On the Algorithmic Structure of Dialectica Realisers

Davide Barbarossa
Department of Computer Science, University of Bath, UK

Thomas Powell
Department of Computer Science, University of Bath, UK

Abstract

Gödel's Dialectica interpretation is a fundamental tool for the extraction of computational content from proofs, and plays a central role in today's proof mining program. In the past decades, it has also been studied from the perspective of programming languages, and our contribution is in that direction. Specifically, we present Dialectica as a collection of rules in the style of Hoare logic, where Dialectica is now viewed as a language for specifying procedural programs that come with a forward and backward direction. This viewpoint captures the interesting dynamics of realisers extracted by the Dialectica interpretation, and we illustrate this by defining a generalised backpropagation semantics for a fragment of this language. We envisage this work as providing a base for several future developments, both theoretical and practical, which we outline at the end.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Constructive mathematics

Keywords and phrases Dialectica interpretation, Hoare logic, Programs from proofs

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.22

Related Version *Full Version*: https://arxiv.org/abs/2501.16208

Funding This work was funded by the Engineering and Physical Sciences Research Council, grant number EP/W035847/1.

Acknowledgements The authors thank Ulrich Berger (who first suggested looking at the frame rule) and Marie Kerjean (who among other things helped clarify several points from [16]).

1 Introduction

Gödel's Dialectica interpretation [12] (which we will often refer to as just "Dialectica") is one of the most important methods for extracting computational content from proofs. Interest in this technique has grown rapidly in recent years, partially due to increased activity in two distinct strands of research.

The first is the proof mining program (an overview of which can be found in the monograph [18] or the survey papers [19, 20]), in which methods from proof theory are applied to obtain both quantitative and qualitative improvements of theorems in mainstream mathematics through the analysis of their proofs. Here, the Dialectica interpretation is used in particular to formulate so-called logical metatheorems, which involve extensions of Dialectica to sophisticated proof systems designed to explain the extractability of quantitative information in a specific setting (see e.g. the classic works [11, 17], or the most recent metatheorem in probability theory [25]).

The second, somewhat orthogonal strand, is represented by a variety of works that approach Dialectica from a programming or categorical point of view, where the focus is on its *structural* properties as a logical and program transformation, rather than on its concrete applications to mathematics. A crucial ingredient here is de Paiva's Dialectica categories and the discovery of the links between Dialectica and linear logic [8] (recent developments of

the categorical viewpoint can be found in e.g. [39]). Most relevant to this paper is the recent reformulation of Dialectica as a genuine higher-order functional program transformation [33], which also led to the discovery of its links with automatic differentiation and the differential λ-calculus [16].

This paper is a study of the Dialectica interpretation that can be related to both of these perspectives: We consider Dialectica in its traditional form as used in proof mining, but with a focus on the *algorithmic structure* of the terms it extracts (referred to here as "Dialectica realisers"). More specifically, we give an alternative presentation of the standard Dialectica interpretation, through a set of rules that that manipulate programs rather than logical formulas. The rules are set up in the style of Hoare logic [14], and act on what we call *Dialectica triples*, which are for now just realizing terms for implications between formulas (the characterising feature of Dialectica), but which we later connect with Hoare triples in the usual sense. Our rules formulate the standard soundness theorem for Dialectica by focusing on properties of extracted realizers, and in this way we seek to expose some of the elegant patterns and symmetries that govern programs extracted by the Dialectica.

We build on this perspective in two ways. First, we introduce a while loop for Dialectica into our term language, and show that it interprets a corresponding rule. We argue that this can be used to neatly describe iterative programs that arise from nonconstructive proofs in mathematics, where it is connected to the idea of interpreting classical proofs via *learning* or *backtracking*, a notion that dates back to Hilbert's substitution method [1].

The fact that Dialectica admits a Hoare Logic presentation or, dually, that Hoare logic admits an extension as a logic for Dialectica realisers, raises the question of whether we can understand them from a procedural perspective. We take a first step in this direction in Section 4, where we consider a restricted set of proof rules whose Dialectica realisers can be regarded as stateful in a suitable sense (though we hesitate to characterise them as truly imperative due to the lack of variable assignment for now, discussed further in Section 5). We describe an operational semantics which demonstrates that our programs perform a generalised backpropagation algorithm, comprising a conventional forward part together with a backward pass that computes the reverse witness.

Our aim in all of this is to provide a perspective that can be properly developed in future work, and in that spirit, we conclude by discussing several concrete ideas, covering both theory and applications.

Our choice of formal system. We will generally privilege a standard presentation of the Dialectica, over more program theoretic ones as found in e.g. [33]. In particular, we work in a higher-order version of Heyting arithmetic rather than a "modern" type theory. Extensions of the former have proven to be more than capable of capturing the application of Dialectica to a great variety of subsystems of ordinary mathematics: An instructive recent example is probability theory, which a-priori requires strong set-theoretic principles to carry out even the most basic tasks, but which is nevertheless amenable to program extraction via the Dialectica as set out in [25] and as exemplified by the numerous case studies that have appeared in the last years. It is open whether alternative systems based on e.g. dependent type theory are better suited to the very precise task of capturing applications of Dialectica to *tame* proofs in ordinary mathematics (see [21] for a discussion of "proof theoretic tameness" in this context), or can be as easily combined with logical relations such as majorizability [15] that are essential for capturing uniformities in extracted programs. Finally, even though as shown in [33] these traditional formulations usually lack a crucial proof/program theoretic property, namely they break β-equivalence, here we embrace this fact at it gives rise to a natural *if-then-else* case distinction construct which fits well with our procedural perspective.

2 Preliminaries

The base system WE-HA$^\omega$

Essentially all applications of Dialectica to concrete proofs in mathematics can be described formally in terms of some theory based on arithmetic in all finite types, and accordingly we take as our base theory the weakly extensional version WE-HA$^\omega$ of higher-order Heyting arithmetic, the higher-order theory of intuitionistic arithmetic with all finite types, whose underlying programming language of terms corresponds to System T. This can be simultaneously used as a system for formalising mathematics, or as a logic for reasoning about higher-order programs, and we take both perspectives in this paper. Full details can be found in [18, Chapter 3], and the very brief overview here is merely used as an opportunity to fix notational conventions. The types of WE-HA$^\omega$ are the simple types

$$X, Y ::= \mathsf{nat} \mid X \to Y$$

where nat represents a base type of natural numbers. As in conventional in proof mining, we work with sequences of types in the metalanguage rather than explicitly introducing product types, which in particular allows us to employ more liberal abbreviations in the logical system. We use boldface $\boldsymbol{X} = X_1, \dots, X_n$ to denote sequences of types, and from now on, when we say "type" we usually refer to a sequence. Similarly, we use boldface $\boldsymbol{a} = a_1, \dots, a_n$ to denote sequences of terms, writing $\boldsymbol{a} : \boldsymbol{X}$ or $\boldsymbol{a}^{\boldsymbol{X}}$ to denote $a_1 : X_1, \dots, a_n : X_n$.

The terms of WE-HA$^\omega$ are those of the simply typed λ-calculus plus constants $0 : \mathsf{nat}$ and $\mathsf{suc} : \mathsf{nat} \to \mathsf{nat}$ for zero and successor (we use $x + 1$ for $\mathsf{suc}\, x$) and, for each sequence $\boldsymbol{X}$ of types, a constant $\mathtt{rec}_{\boldsymbol{X}} : (\mathsf{nat} \to \boldsymbol{X} \to \boldsymbol{X}) \to \boldsymbol{X} \to \mathsf{nat} \to \boldsymbol{X}$ for primitive recursion. We include an explicit cases constructor $\mathtt{if}(b^{\mathsf{nat}}, \boldsymbol{s}, \boldsymbol{t})$ for all types, even though this is definable from the recursor. We make free use of a number of standard abbreviations around sequences of types and terms. If $\boldsymbol{X} = X_1, \dots, X_n$ and $\boldsymbol{Y} = Y_1, \dots, Y_m$ then $\boldsymbol{X} \to \boldsymbol{Y}$ denotes the sequence $(X_1 \to \dots \to X_n \to Y_j)_{j=1}^m$, Similarly, if $\boldsymbol{a} : \boldsymbol{X} \to \boldsymbol{Y}$ and $\boldsymbol{b} : \boldsymbol{X}$, then by $\boldsymbol{ab} : \boldsymbol{Y}$ we mean $(a_j b_1 \dots b_n)_{j=1}^m$, and if $\boldsymbol{c} : \boldsymbol{Y}$ are terms and $\boldsymbol{x} : \boldsymbol{X}$ are variables, then by $\lambda \boldsymbol{x}.\boldsymbol{c} : \boldsymbol{X} \to \boldsymbol{Y}$ we mean$(\lambda x_1, \dots, x_n.c_j)_{j=1}^m$. We write $\boldsymbol{b}, \boldsymbol{c} : \boldsymbol{X}, \boldsymbol{Y}$ for the concatenation of two sequences.

Formulas of WE-HA$^\omega$ are built from atomic formulas of the form $t =_{\mathsf{nat}} s$ for $t, s : \mathsf{nat}$, the usual logical connectives $\vee, \wedge, \to, \top, \bot$ (where we write $\neg A := A \to \bot$), and quantifiers $\forall^X, \exists^X$ for each simple type X (which we usually omit). We also write $\exists \boldsymbol{x}\, A(\boldsymbol{x})$ for $\exists^{X_1} x_1, \dots, \exists^{X_n} x_n\, A(x_1, \dots, x_n)$ and similarly for $\forall$. Equality $=_X$ at higher types is defined in terms of $=_{\mathsf{nat}}$, where for $X = X_1 \to \dots \to X_n \to \mathsf{nat}$ we set

$$t =_X s := \forall x_1, \dots, x_n (t x_1 \dots x_n =_{\mathsf{nat}} s x_1 \dots x_n).$$

Equality for sequences is defined in the obvious way. The axioms and rules of WE-HA$^\omega$ are those of usual intuitionistic logic extended to all simple types, along with equality axioms for $=_{\mathsf{nat}}$, induction, and axioms for the arithmetical constants and terms of System T. For instance, the conditional satisfies

$$b =_{\mathsf{nat}} 0 \to \mathtt{if}(b, \boldsymbol{s}, \boldsymbol{t}) = \boldsymbol{s} \quad \text{and} \quad b \neq_{\mathsf{nat}} 0 \to \mathtt{if}(b, \boldsymbol{s}, \boldsymbol{t}) = \boldsymbol{t}$$

and the axioms for the recursors are:

$$\mathtt{rec}\, \boldsymbol{z}\boldsymbol{y} 0 = \boldsymbol{y} \quad \text{and} \quad \mathtt{rec}\, \boldsymbol{z}\boldsymbol{y}(x+1) = \boldsymbol{z} x (\mathtt{rec}\, \boldsymbol{z}\boldsymbol{y} x).$$

For equality, we consider the weak extensionality rule[1]

$$\frac{A_0 \to a =_X b}{A_0 \to B[a/x] \to B[b/x]}$$

where B is an arbitrary formula and A_0 is quantifier free. For full details of WE-HA$^\omega$ the reader is directed to [18], though exact details at this level are unimportant for what follows.

The Dialectica interpretation

We give a brief overview of the Dialectica interpretation of *formulas* (as opposed to *proofs*). This is completely standard, and full details can be found in [18, Chapter 8]. In simple terms, the Dialectica interpretation is a mapping $P \mapsto \exists \boldsymbol{x} \forall \boldsymbol{y} |P|^{\boldsymbol{x}}_{\boldsymbol{y}}$ that transforms formulas from some source logic (originally Heyting arithmetic) to some higher-order functional language (originally System T). The transformed formula $|P|^{\boldsymbol{x}}_{\boldsymbol{y}}$ contains two additional tuples of free variables, playing the role of witnesses $\boldsymbol{x}$ and counterwitnesses $\boldsymbol{y}$ for P, while the formula $|P|^{\boldsymbol{x}}_{\boldsymbol{y}}$ can be viewed as an *orthogonality relation* similarly to classical realisability. The aim is to establish that whenever P is provable in the source system, there exists terms $\boldsymbol{a}$ such that $\forall \boldsymbol{y} |P|^{\boldsymbol{a}}_{\boldsymbol{y}}$ is provable in the target system. For a term $t : \mathsf{nat}$, we introduce a new abbreviation $\vee_t$ for the formula

$$A \vee_t B := (t =_{\mathsf{nat}} 0 \to A) \wedge (t \neq_{\mathsf{nat}} 0 \to B)$$

Using the fact that any quantifier-free formula ϕ of WE-HA$^\omega$ can be represented by a "characteristic term" $\chi_\phi : \mathsf{nat}$ with the same free variables satisfying $\chi_\phi = 0 \leftrightarrow \phi$, we extend this connective to such formulas by defining $A \vee_\phi B := A \vee_{\chi_\phi} B$.

▶ **Definition 1** (Dialectica interpretation [12], see also [18]). *For a formula P of* WE-HA$^\omega$*, we define its Dialectica interpretation to be the formula $\exists \boldsymbol{x} \forall \boldsymbol{y} \, |P|^{\boldsymbol{x}}_{\boldsymbol{y}}$, whose free variables are the same as P, and where $|P|^{\boldsymbol{x}}_{\boldsymbol{y}}$ is a quantifier-free formula, defined inductively along with the types of $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows:*

- $|P| := P$ *if P is atomic*
- $|P \wedge Q|^{\boldsymbol{x},\boldsymbol{u}}_{\boldsymbol{y},\boldsymbol{v}} := |P|^{\boldsymbol{x}}_{\boldsymbol{y}} \wedge |Q|^{\boldsymbol{u}}_{\boldsymbol{v}}$
- $|P \vee Q|^{b,\boldsymbol{x},\boldsymbol{u}}_{\boldsymbol{y},\boldsymbol{v}} := |P|^{\boldsymbol{x}}_{\boldsymbol{y}} \vee_b |Q|^{\boldsymbol{u}}_{\boldsymbol{v}}$
- $|P \to Q|^{\boldsymbol{f},\boldsymbol{F}}_{\boldsymbol{x},\boldsymbol{v}} := |P|^{\boldsymbol{x}}_{\boldsymbol{F}\boldsymbol{x}\boldsymbol{v}} \to |Q|^{\boldsymbol{f}\boldsymbol{x}}_{\boldsymbol{v}}$
- $|\exists x \, P(x)|^{x,\boldsymbol{u}}_{\boldsymbol{y}} := |P(x)|^{\boldsymbol{u}}_{\boldsymbol{y}}$
- $|\forall x \, P(x)|^{\boldsymbol{f}}_{x,\boldsymbol{y}} := |P(x)|^{\boldsymbol{f}x}_{\boldsymbol{y}}$

Note that if ϕ is quantifier-free then we can assume that it doesn't contain $\vee$ (as it can be rewritten as $\chi_\phi = 0$) and therefore that $|\phi| = \phi$.

The characteristic feature of Dialectica, which sets it apart from realisability-like interpretations, is its treatment of implication, which comprises a forward-component $\boldsymbol{f}$ as in realisability along with a backward component $\boldsymbol{F}$, which together compose in a "backpropagation" style. The soundness theorem for the Dialectica interpretation (i.e. the Dialectica interpretation of *proofs*), for extensions of WE-HA$^\omega$, usually takes the following general form:

[1] We stress that extensionality issues are not a concern for what we do: Soundness theorems for Dialectica are formulated using WE-HA$^\omega$ for the simple reason that the extensionality *axiom* is not admissible. However, because our focus in on a descriptive system for realizers, we could equally well work in the fully extensional version of Heyting arithmetic E-HA$^\omega$.

▶ **Theorem 2** (Generic soundness theorem). *Suppose that Δ is a set of axioms whose Dialectica interpretation is witnessed by terms of* WE-HA$^\omega_\Delta$ *(provably in this system) for some suitable extension of* WE-HA$^\omega$*, and let $\mathcal{U}$ be a set of purely universal formulas. Then whenever*

$$\text{WE-HA}^\omega + \Delta + \mathcal{U} \vdash P$$

we can extract terms $\boldsymbol{a}$, whose free variables are the same as those of P, such that

$$\text{WE-HA}^\omega_\Delta + \mathcal{U} \vdash \forall \boldsymbol{y} |P|^{\boldsymbol{a}}_{\boldsymbol{y}}.$$

In particular, we can set $\Delta = \mathcal{U} = \emptyset$ to obtain the Dialectica interpretation of WE-HA$^\omega$. A common formulation (cf. [18, Theorem 8.6]) has Δ consisting of the axiom of choice (AC), the independence of premise scheme for universal formulas (IP$^\omega_\forall$), and Markov's principle (M$^\omega$) with WE-HA$^\omega_\Delta$ = WE-HA$^\omega$. A further extension is to add the double negation shift to Δ, and then WE-HA$^\omega_\Delta$ = WE-HA$^\omega$ + (BR) i.e. Heyting arithmetic extended with bar recursion in all finite types (cf. [18, Chapter 11]).

3 The action of Dialectica on programs

We now present the soundness theorem for the Dialectica interpretation from a slightly different perspective, by providing a collection of specification rules for Dialectica realizers (terminology we use for programs extracted from proofs using Dialectica), all of which are sound in our base theory WE-HA$^\omega$. The intention here is to construct a rich, descriptive language where the focus is on the realizing terms rather than the underlying logic, one which can be easily extended with new programming constructs, or new rules for describing extracted programs of a specific type. We give several examples of this in what follows.

Our basic approach is inspired by Hoare logic [14], in the sense that our rules apply to "triples" that describe properties of Dialectica realizers. Let us explain this in more detail. Viewed abstractly, a basic Hoare triple $\{P\}\, c\, \{Q\}$ describes the effect of some command c in an imperative language on an underlying state. Here P (the precondition) and Q (the postcondition) are assertions about the state, and the meaning of the triple is that whenever the command executes on a state satisfying P, the resulting state satisfies Q. We can represent this basic idea in predicate logic by viewing the pre- and postconditions as formulas $P(s), Q(s)$ acting on objects $s : S$ for some state type S, and commands as functions $c : S \to S$. The Hoare triple then becomes the statement that c satisfies

$$\forall s (P(s) \to Q(cs)).$$

If we now imagine that $P(s)$ and $Q(s)$ are quantifier-free formulas, and the state can be encoded in a suitable way in WE-HA$^\omega$, the formula $P(s) \to Q(cs)$ is exactly

$$|\exists x\, P(x) \to \exists x\, Q(x)|^c.$$

With this loose correspondence in mind, we now generalise to formulas of arbitrary logical complexity and define a "Dialectica triple" as

$$P\, \langle \boldsymbol{a} \,|\, \boldsymbol{\alpha} \rangle\, Q := \forall \boldsymbol{x}, \boldsymbol{v}\, |P \to Q|^{\boldsymbol{a}, \boldsymbol{\alpha}}_{\boldsymbol{x}, \boldsymbol{v}}.$$

where now $\boldsymbol{a}$ and $\boldsymbol{\alpha}$ are sequences of terms of some type determined by the formulas P and Q. In terms of actions on a hypothetical state $\boldsymbol{x}$, we visualise the pair $\langle \boldsymbol{a} \,|\, \boldsymbol{\alpha} \rangle$ as comprising two components: A forward "command" $\boldsymbol{a}$ satisfying

$$\forall \boldsymbol{x} \left(\forall \boldsymbol{y} |P|^{\boldsymbol{x}}_{\boldsymbol{y}} \to \forall \boldsymbol{v} |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \right)$$

Propositional rules: Axioms, basic actions, conditionals, switching, composition.

$$\bot\,\langle \boldsymbol{a} \mid - \rangle\, P \qquad P\,\langle - \mid \boldsymbol{\alpha} \rangle\, \top \qquad P\,\langle \lambda\boldsymbol{x}.\boldsymbol{x} \mid \lambda\boldsymbol{x},\boldsymbol{v}.\boldsymbol{v} \rangle\, P \qquad \frac{P_\exists \to Q_\forall \in \mathrm{Ax}}{P_\exists\,\langle - \mid - \rangle\, Q_\forall} \qquad \frac{P_\exists\,\langle - \mid - \rangle\, Q_\forall}{P'_\exists\,\langle - \mid - \rangle\, Q'_\forall} \text{ for } \frac{P_\exists \to Q_\forall}{P'_\exists \to Q'_\forall} \in \mathrm{Rule}$$

$$\frac{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, Q \wedge R}{P\,\langle \boldsymbol{b},\boldsymbol{a} \mid \tilde{\boldsymbol{\alpha}} \rangle\, R \wedge Q}\,{}^{p\wedge}{}_R \qquad \frac{P \wedge Q\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha},\boldsymbol{\beta} \rangle\, R}{Q \wedge P\,\langle \tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\alpha}} \rangle\, R}\,{}^{p\wedge}{}_L \qquad \frac{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, Q \vee_c R}{P\,\langle \boldsymbol{b},\boldsymbol{a} \mid \tilde{\boldsymbol{\alpha}} \rangle\, R \vee_{\tilde{c}} Q}\,{}^{p\vee}{}_R \qquad \frac{P \vee_c Q\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha},\boldsymbol{\beta} \rangle\, R}{Q \vee_{\tilde{c}} P\,\langle \tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\alpha}} \rangle\, R}\,{}^{p\vee}{}_L$$

$$\frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q}{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha}_\pi \rangle\, Q \vee_0 R}\,{}^{\vee}{}_R \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q}{P \wedge R\,\langle \boldsymbol{a}_\pi \mid \boldsymbol{\alpha}_\pi,\boldsymbol{\beta} \rangle\, Q}\,{}^{\wedge}{}_L \qquad \frac{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, Q \wedge R}{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}_p \rangle\, Q}\,{}^{\wedge}{}_R \qquad \frac{P \vee_0 R\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha},\boldsymbol{\beta} \rangle\, Q}{P\,\langle \boldsymbol{a}_p \mid \boldsymbol{\alpha}_p \rangle\, Q}\,{}^{\vee}{}_L$$

$$\frac{P \wedge \phi\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, R \quad Q \wedge \neg\phi\,\langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle\, R}{P \vee_\phi Q\,\langle \lambda\boldsymbol{x},\boldsymbol{y}.\mathtt{if}(\phi,\boldsymbol{a}\boldsymbol{x},\boldsymbol{b}\boldsymbol{y}) \mid \boldsymbol{\alpha}_\pi,\boldsymbol{\beta}_\pi \rangle\, R}\,{}^{cond}{}_L \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q \quad P\,\langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle\, R}{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \lambda\boldsymbol{x},\boldsymbol{v},\boldsymbol{w}.\mathtt{if}(|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}},\boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w},\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}) \rangle\, Q \wedge R}\,{}^{cond}{}_R$$

$$\frac{P\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, Q \to R}{P \wedge Q\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha},\boldsymbol{b} \rangle\, R}\,{}^{imp} \qquad \frac{P \wedge Q\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha},\boldsymbol{\beta} \rangle\, R}{P\,\langle \boldsymbol{a},\boldsymbol{\beta} \mid \boldsymbol{\alpha} \rangle\, Q \to R}\,{}^{exp} \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q \quad Q\,\langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle\, R}{P\,\langle \boldsymbol{b} \circ \boldsymbol{a} \mid \boldsymbol{\alpha} *_{\boldsymbol{a}} \boldsymbol{\beta} \rangle\, R}\,{}^{comp}$$

Quantifier rules: Term introduction, λ-abstraction and application, epsilon terms.

$$\frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q(\boldsymbol{t})}{P\,\langle \lambda\_.\boldsymbol{t},\boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, \exists\boldsymbol{x}\, Q(\boldsymbol{x})}\,{}^{\exists}{}_R \qquad \frac{P(\boldsymbol{t})\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q}{\forall\boldsymbol{x}\, P(\boldsymbol{x})\,\langle \lambda\boldsymbol{f}.\boldsymbol{a}(\boldsymbol{f}\boldsymbol{t}) \mid \lambda\_.\boldsymbol{t},\lambda\boldsymbol{f}.\boldsymbol{\alpha}(\boldsymbol{f}\boldsymbol{t}) \rangle\, Q}\,{}^{\forall}{}_L$$

$$\frac{P(\boldsymbol{x})\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q}{\exists\boldsymbol{x}\, P(\boldsymbol{x})\,\langle \lambda\boldsymbol{x}.\boldsymbol{a} \mid \lambda\boldsymbol{x}.\boldsymbol{\alpha} \rangle\, Q}\,{}^{\exists}{}_L \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q(\boldsymbol{x})}{P\,\langle \lambda\boldsymbol{y},\boldsymbol{x}.\boldsymbol{a}\boldsymbol{y} \mid \lambda\boldsymbol{y},\boldsymbol{x}.\boldsymbol{\alpha}\boldsymbol{y} \rangle\, \forall\boldsymbol{x}\, Q(\boldsymbol{x})}\,{}^{\forall}{}_R \qquad \boldsymbol{x} \text{ not free in } Q \text{ resp. } P \text{ for } \exists_L \text{ resp. } \forall_R$$

$$\frac{\exists\boldsymbol{x}\, P(\boldsymbol{x})\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q}{P(\boldsymbol{t})\,\langle \boldsymbol{a}\boldsymbol{t} \mid \boldsymbol{\alpha}\boldsymbol{t} \rangle\, Q}\,{}^{s}{}_L \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, \forall\boldsymbol{x}\, Q(\boldsymbol{x})}{P\,\langle \lambda\boldsymbol{y}.\boldsymbol{a}\boldsymbol{y}\boldsymbol{t} \mid \lambda\boldsymbol{y},\boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{y}\boldsymbol{t}\boldsymbol{v} \rangle\, Q(\boldsymbol{t})}\,{}^{s}{}_R \qquad \frac{P_\forall\,\langle \boldsymbol{a},\boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, \exists\boldsymbol{x}\, Q(\boldsymbol{x})}{P_\forall\,\langle \boldsymbol{b} \mid \boldsymbol{\alpha} \rangle\, Q(\boldsymbol{a})}\,{}^{\epsilon}{}_R \qquad \frac{\forall\boldsymbol{x}\, P_\forall(\boldsymbol{x})\,\langle - \mid \boldsymbol{\alpha},\boldsymbol{\beta} \rangle\, Q_{qf}}{P_\forall(\boldsymbol{\alpha})\,\langle - \mid \boldsymbol{\beta} \rangle\, Q_{qf}}\,{}^{\epsilon}{}_L$$

Consequence, extensionality, and induction/recursion

$$\frac{P' \to_D P \quad P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q \quad Q \to_D Q'}{P'\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q'}\,{}^{cons} \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle\, Q \quad \boldsymbol{a},\boldsymbol{\alpha} = \boldsymbol{b},\boldsymbol{\beta}}{P\,\langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle\, Q}\,{}^{ext} \qquad \frac{P(x)\,\langle \boldsymbol{a}(x) \mid \boldsymbol{\alpha}(x) \rangle\, P(x+1)}{P(0)\,\langle \mathtt{rec}\,\boldsymbol{a} \mid \mathtt{rec}^*\,\boldsymbol{a}\boldsymbol{\alpha} \rangle\, \forall x\, P(x)}\,{}^{ind}$$

Figure 1 Rules for Dialectica triples, definitions and abbreviations discussed in Section 3.1.

followed by a backward command $\boldsymbol{\alpha}$, that operates on a "dual state" $\boldsymbol{v}$, satisfying

$$\forall\boldsymbol{x},\boldsymbol{v}\,(\neg|Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \to \neg|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}})$$

This approach differs from the use of Hoare triples for realizability in [36], which is based on a monadic translation of proofs: Here Hoare triples are just the interpretation of implication, and a corresponding stateful interpretation in terms of backpropagation will be discussed in more detail in Section 4 below.

As we will see, our use of Hoare logic *in the abstract* allows us to view Dialectica realisers as "stateful" or *procedural* programs. However, we are cautious in drawing too strong a connection: The fundamental power of Hoare logic and its various extensions lies in its ability to reason explicitly about the structure of the state, something that in connection to Dialectica we leave to future work. Therefore at present, we do not characterise Dialectica realisers as truly *imperative*. However, we do believe that such view is possible and we think of this work as the first step towards it.

3.1 A Dialectica Hoare logic (DHL)

It is natural to ask what Hoare-like rules govern Dialectica realizers when viewed in this way. Fixing sets $\mathrm{Ax}, \mathrm{Rule}$ of axioms and rules in WE-HA$^\omega$, we define a judgment system DHL (for *Dialectica Hoare Logic*) which derives judgments of shape $P\langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle Q$, where P, Q are formulas of WE-HA$^\omega$ and $\boldsymbol{a}, \boldsymbol{\alpha}$ are (sequences of) terms of System T, according to the rules in Figure 1. In all cases, the types of $\boldsymbol{a}, \boldsymbol{\alpha}$ are left implicit, but can be inferred from the rules and formulas, as usual in Dialectica. These rules are described in detail in the points below, where we also clarify the notations and abbreviations used.

- We write $P_\exists$ to denote a formula whose Dialectica interpretation is purely existential i.e. of the form $\exists \boldsymbol{x}\, |P|^{\boldsymbol{x}}$. Similarly, $P_\forall$ and P_{qf} mean that the Dialectica interpretation of P is purely universal resp. quantifier-free.
- The sets Ax and Rule denote sets of universal axioms and rules that we add to the system: At the very least Ax would include the axioms and rules for $=_{\mathsf{nat}}$ along with those governing System T terms, while Rule would include the quantifier-free extensionality rule, but we leave open the possibility that other axioms and rules could be included.
- For $c : \mathsf{nat}$ we define $\bar{c}$ by $\bar{0} := 1$ and $\overline{n+1} := 0$.
- $\tilde{\boldsymbol{\alpha}}$ denotes a permutation of the arguments of $\boldsymbol{\alpha}$, whose precise definition varies but can be immediately inferred from the rule. For example, written out fully, the $\tilde{\boldsymbol{\alpha}}$ in the conclusion of $(p\wedge_R)$ should be $\tilde{\boldsymbol{\alpha}} := \lambda \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}\boldsymbol{w}$ and the intuitive meaning of the triple is the formula $\forall \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{v}\, \left(|P|^{\boldsymbol{x}}_{\tilde{\boldsymbol{\alpha}}\boldsymbol{x}\boldsymbol{w}\boldsymbol{v}} \to |R|^{\boldsymbol{b}\boldsymbol{x}}_{\boldsymbol{w}} \wedge |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}}\right)$. This further illustrates our philosophy on implicit typing: The aim of our notation is to suppress bureaucratic λ-terms wherever these can be directly inferred.
- In a similar way, $\boldsymbol{\alpha}_\pi$ denotes a projection (e.g. in $(\vee_R)$, $\boldsymbol{\alpha}_\pi$ is shorthand for $\lambda \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}$), while $\boldsymbol{\alpha}_p$ denotes a coprojection, by which we mean the instantiation of certain arguments with canonical zero terms $\mathbf{0}$ of the right type (e.g. in $(\wedge_R)$, $\boldsymbol{\alpha}_p$ is shorthand for $\lambda \boldsymbol{x}, \boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}\mathbf{0}$).
- $\lambda\_.t$ denotes the constant term $\lambda \boldsymbol{x}.\boldsymbol{t}$, where the types of $\boldsymbol{x}$, not free in t, are to be inferred.
- For the conditional rules, ϕ is always quantifier-free.
- For composition, $\circ$ denotes the usual composition of functions, while $*$ denotes the special backwards composition for the Dialectica interpretation, with

$$\boldsymbol{\alpha} *_{\boldsymbol{a}} \boldsymbol{\beta} := \lambda \boldsymbol{x}, \boldsymbol{w}.\boldsymbol{\alpha}\boldsymbol{x}\,(\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{w})$$

- For the quantifier rules we write $Q(\boldsymbol{x})$ to denote all occurrences of the free variables $\boldsymbol{x}$ in Q, and note that there may be no occurrences. We assume the usual rules for substitution e.g. in $(\exists_R)$ that $\boldsymbol{t}$ is free for $\boldsymbol{x}$ in Q, and similarly for the other rules.
- Rule (*cons*) is the Dialectica version of the usual consequence rules, where the pre- and postconditions can be weakened/strengthened with no bearing on the program. More precisely, we define the WE-HA$^\omega$-formula $P' \to_D P$ by

$$\forall \boldsymbol{x}, \boldsymbol{v}\, (|P'|^{\boldsymbol{x}}_{\boldsymbol{v}} \to |P|^{\boldsymbol{x}}_{\boldsymbol{v}})$$

  i.e. the witnessing types of P' and P coincide under the Dialectica and implication is interpreted by the identity.
- Finally, $\mathtt{rec}$ denotes the usual recursor whose defining axiom is given in Section 2, while the backward recursor $\mathtt{rec}^*$ is defined (using the main recursor) as $\mathtt{rec}^*\boldsymbol{a}\boldsymbol{\alpha}\boldsymbol{y}x\boldsymbol{v} := \boldsymbol{\beta}x$ for

$$\boldsymbol{\beta}0 := \boldsymbol{v} \quad \text{and} \quad \boldsymbol{\beta}(z+1) := \boldsymbol{\alpha}(x-z-1)(\mathtt{rec}\,\boldsymbol{a}\boldsymbol{y}(x-z-1))(\boldsymbol{\beta}z).$$

▶ **Theorem 3.** *Assuming that all elements of* Ax *and* Rule *are admissible in* WE-HA$^\omega$*, if* $P\,\langle \boldsymbol{a} \,|\, \boldsymbol{\alpha}\rangle\, Q$ *is derivable from the rules in Figure 1, then* $\forall \boldsymbol{x}, \boldsymbol{v}\, (|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \to |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}})$ *is provable in* WE-HA$^\omega$.

The proof of Theorem 3 is routine, and involves no ideas fundamentally different to those already present in the usual soundness proofs of Dialectica for WE-HA$^\omega$. Some of the more interesting rules are discussed in the appendix. We note that by restricting the rules of DHL so that the precondition is always purely existential, we obtain a simplified set of rules that act only on left-hand realizers, and very closely reflect traditional Hoare rules. These are also presented in the appendix, in Figure 9.

$$\frac{P \vee Q \,\langle \boldsymbol{a} \,|\, \boldsymbol{\alpha}, \boldsymbol{\beta}\rangle\, R}{Q \vee P \,\langle \lambda c.\tilde{\boldsymbol{a}}\bar{c} \,|\, \lambda c.\tilde{\boldsymbol{\beta}}\bar{c}, \lambda c.\tilde{\boldsymbol{\alpha}}\bar{c}\rangle\, R}\, p\vee'_L \qquad \frac{P \,\langle \boldsymbol{a} \,|\, \boldsymbol{\alpha}\rangle\, Q}{P \,\langle \lambda \boldsymbol{x}.0, \boldsymbol{a}, \boldsymbol{b} \,|\, \boldsymbol{\alpha}_\pi\rangle\, Q \vee R}\, \vee'_R$$

$$\frac{P \,\langle \boldsymbol{a} \,|\, \boldsymbol{\alpha}\rangle\, R \quad Q \,\langle \boldsymbol{b} \,|\, \boldsymbol{\beta}\rangle\, R}{P \vee Q \,\langle \lambda c, \boldsymbol{x}, \boldsymbol{y}.\mathtt{if}(c = 0, \boldsymbol{a}\boldsymbol{x}, \boldsymbol{b}\boldsymbol{y}) \,|\, \alpha_\pi, \beta_\pi\rangle\, R}\, cond'_L$$

Figure 2 Rules for disjunction, derivable from those in Figure 1, which can be added to DHL.

Using DHL

Our system DHL is, above all, intended to be a useful system for constructing and describing realizing terms. It is certainly not a minimal system: Indeed, because (*cons*) ranges over all implications of form $\to_D$ derivable in WE-HA$^\omega$, it can be used together with the quantifier introduction rules to derive all of the others. However, a human (or a machine) using the system would typically use (*cons*) only in in cases where $P' \to_D P$ and $Q \to_D Q'$ are immediate (e.g. have no computational content), and would resort to the main rules for constructing complex terms.

We view the system as fundamentally extendible: The rules in Figure 1 provide the basic manipulations on terms required to show that the Dialectica is sound for WE-HA$^\omega$, but in practice, especially if we envisage DHL being incorporated into some proof assistant as a way of reasoning about Dialectica realizers, we would add a range of derivable rules for dealing with specific mathematical structures in proofs, along with new rules for characterising additional constructs added to System T. We give several examples of these in what follows. As a simple illustration, in Figure 2 we note several rules involving disjunction that are easily derivable from those for $\vee_b$ along with (*cons*) and the fact that $P \vee Q \leftrightarrow_D \exists b(P \vee_b Q)$.

It is now natural to ask: *What purely logical system is mirrored by* DHL*?* This can be quite interesting, especially so when we consider extensions to our rules later on. We first note that if we remove (ϵ_L), (ϵ_R), (*cons*), (*ext*) from those of Figure 1, and then replace the disjunction rules with those of Figure 2, these rules can be adapted to form a logical system in the language of WE-HA$^\omega$ by just suppressing the realizing term. For example, we can simplify the rule ($cond'_L$) as

$$\frac{P \vdash R \quad Q \vdash R}{P \vee Q \vdash R}\, cond'_L$$

and similarly for the others. Letting $\mathcal{H}$ denote the resulting system, we have the following:

▶ **Theorem 4.** *Whenever P is provable in* WE-HA$^\omega$*, then $\top \vdash P$ is provable in $\mathcal{H}$.*

Again, the proof is straightforward, and a rough sketch of how we show this is provided in the appendix. Theorem 4 therefore represents an alternative soundness proof from the Dialectica interpretation of WE-HA$^\omega$, and another way of using DHL: If P is provable in WE-HA$^\omega$, we take an alternative derivation of $\top \vdash P$ in $\mathcal{H}$ and adding back the hidden realizing terms we obtain a derivation of

$$\top \,\langle \boldsymbol{a} \,|\, -\rangle\, P = \forall \boldsymbol{v} |P|^{\boldsymbol{a}}_{\boldsymbol{v}}$$

where $\boldsymbol{a}$ is constructed via the derivation (note that we can also force that the free variables of $\boldsymbol{a}$ are contained in those of P by applying ($\exists_L$) on all superfluous variables and then (s_L) instantiated with zero terms). We also observe that if $P \to Q$ is any formula such that

$P \leftrightarrow_D R \leftrightarrow_D Q$ for some suitable R, then $P \langle \lambda \boldsymbol{x}.\boldsymbol{x} \mid \lambda \boldsymbol{x}, \boldsymbol{v}.\boldsymbol{v} \rangle Q$ is immediately derivable from the (*cons*) rule and therefore $P \vdash Q$ can be added to $\mathcal{H}$. In this way, we also regain that AC, $\mathrm{IP}^{\omega}_{\forall}$ and M^{ω} are admissible by the Dialectica.

There are still three rules we have not discussed. While (*ext*) is just an extensionality rule purely for reasoning about extracted terms, (ϵ_L) and (ϵ_R) are more interesting. These essentially allow us access to *epsilon terms*: Informally speaking, we can add these rules to our purely logical system $\mathcal{H}$ as

$$\frac{P_\forall \vdash \exists \boldsymbol{x}\, Q(\boldsymbol{x})}{P_\forall \vdash Q(\boldsymbol{\epsilon})}\,\epsilon_R \qquad \frac{\forall \boldsymbol{x}\, P_\forall(\boldsymbol{x}) \vdash Q_{qf}}{P_\forall(\boldsymbol{\epsilon}) \vdash Q_{qf}}\,\epsilon_L$$

where here we would add a countable set of epsilon terms (ϵ_i) to our language and use a fresh sequence of ϵ-terms for each instance of (ϵ_L), (ϵ_R) in a derivation. The hidden realizing terms allow us to always assign concrete values to these ϵ-terms, which can then be substituted in should any remain at the end of a derivation. However, some care is needed to set this up formally. Interestingly, the addition of ϵ-terms to *intuitionistic* logic is explored in [23]: There, doing this for arbitrary formulas does not give us a conservative extension of intuitionistic logic, as the full independence of premise axiom is then derivable[2]. Our in-built restriction for (ϵ_R) that the left hand formula has purely universal Dialectica interpretation avoids this problem: We are then only able to derive independence of premise for *universal* formulas, and this is admissible by Dialectica!

Finally, we note that by taking advantage of the infrastructure is already available via Oliva's unified approach to functional interpretations [30], it would not be hard to parametrise our own system DHL with abstract bounding relation $\sqsubset$ and thereby obtain both the traditional Dialectica, modified realizability and the Diller-Nahm interpretation [10] as instances. For the latter, the conditional rule $(cond_R)$ would then look like

$$\frac{P \langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle Q \quad P \langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle R}{P \langle \boldsymbol{a}, \boldsymbol{b} \mid \lambda \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}.\, (\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v} \cup \boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w}) \rangle\, Q \wedge R}\, cond_R$$

for $\cup$ a union operation on finite sets.

3.2 A while rule for Dialectica, and its use in classical logic

We characterised DHL as being inspired by Hoare logic, but conspicuously absent so far are any kind of imperative control flow statements. To that end, we introduce a while construct and identify a corresponding rule that is sound in our base system. We argue that in the context of Dialectica, this while loop is well-suited to describing how programs extracted from classical logic implement "learning algorithms".

For the present paper, we restrict our attention to programs that are *total*, so our while rule is specified relative to some wellfounded relation. We first introduce a decidable binary relation $\prec$ on objects of type $\boldsymbol{X}$, which can be modelled in WE-HA^{ω} as a function $\prec\colon \boldsymbol{X} \to \boldsymbol{X} \to \mathsf{nat}$. We then expand WE-$\mathrm{HA}^{\omega}$ with a wellfounded induction rule $\mathrm{I}_{\prec}$ given by

$$\frac{\forall \boldsymbol{x}\, (\forall \boldsymbol{y} \prec \boldsymbol{x}\, A(\boldsymbol{y}) \to A(\boldsymbol{x}))}{\forall \boldsymbol{x}\, A(\boldsymbol{x})}\,\mathrm{I}_{\prec}$$

[2] We are grateful to Cameron Allett for directing us to [23].

$$\frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}\rangle\,\neg\neg Q}{\neg Q\,\langle \tilde{\boldsymbol{\alpha}} \mid \tilde{\boldsymbol{a}}\rangle\,\neg P}\,CP \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}\rangle\,\forall \boldsymbol{g}\exists \boldsymbol{u}\,|Q|^{\boldsymbol{u}}_{\boldsymbol{gu}}}{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}\rangle\,\neg\neg Q}\,N \qquad \frac{P\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}\rangle\,\neg\neg Q \quad Q\,\langle \boldsymbol{b} \mid \boldsymbol{\beta}\rangle\,R_\exists}{P\,\langle \lambda \boldsymbol{x}.\boldsymbol{b}(\boldsymbol{ax}\boldsymbol{\beta}) \mid \lambda \boldsymbol{x}.\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{\beta}\rangle\,R_\exists}\,comp_\neg$$

Figure 3 Example rules for manipulating classical proofs under the double negation translation.

where $A(\boldsymbol{x})$ ranges over arbitrary formulas. Now, for a quantifier-free formula $\phi(\boldsymbol{x}^{\boldsymbol{X}})$ and a term $\boldsymbol{a} : \boldsymbol{X} \to \boldsymbol{X}$, for any sequence of types $\boldsymbol{U}$ we add to the language of WE-HA$^\omega$ a while recursor $\mathtt{whilerec}_{\prec,\phi,\boldsymbol{a}} : (\boldsymbol{X} \to \boldsymbol{U}) \to (\boldsymbol{X} \to \boldsymbol{U} \to \boldsymbol{U}) \to \boldsymbol{X} \to \boldsymbol{U}$ along with the schema

$$(\phi(\boldsymbol{x}) \to \boldsymbol{ax} \prec \boldsymbol{x}) \to \mathtt{whilerec}_{\prec,\phi,\boldsymbol{a}}\boldsymbol{uFx} =_{\boldsymbol{U}} \mathtt{if}(\phi(\boldsymbol{x}), \boldsymbol{Fx}\,(\mathtt{whilerec}_{\prec,\phi,\boldsymbol{a}}\boldsymbol{uF}(\boldsymbol{ax})), \boldsymbol{ux})$$

where here the premise ensures that triggering the condition of the while loop causes a descent along $\prec$. We write WE-HA$^\omega + (\mathtt{whilerec}_\prec)$ for WE-HA$^\omega$ extended with the while recursor and its axiom for all ϕ and $\boldsymbol{a}$. Now we define the forward while operator $\mathtt{while}^+_\prec \phi\,\mathtt{do}\,\boldsymbol{a} : \boldsymbol{X} \to \boldsymbol{X}$ as

$$\mathtt{while}^+_\prec \phi\,\mathtt{do}\,\boldsymbol{a} := \mathtt{whilerec}_{\prec,\phi,\boldsymbol{a}}(\lambda \boldsymbol{x}.\boldsymbol{x})(\lambda \boldsymbol{x}, \boldsymbol{y}.\boldsymbol{y})$$

and the backward operator $\mathtt{while}^-_\prec \phi\,\mathtt{do}\,(\boldsymbol{a}, \boldsymbol{\alpha}) : \boldsymbol{X} \to \boldsymbol{V} \to \boldsymbol{V}$ for $\boldsymbol{\alpha} : \boldsymbol{X} \to \boldsymbol{V} \to \boldsymbol{V}$ as

$$\mathtt{while}^-_\prec \phi\,\mathtt{do}\,(\boldsymbol{a}, \boldsymbol{\alpha}) := \mathtt{whilerec}_{\prec,\phi,\boldsymbol{a}}(\lambda \boldsymbol{x}, \boldsymbol{v}.\boldsymbol{v})(\lambda \boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{x}(\boldsymbol{fv})).$$

The following result, easily proven by expanding the definitions, guarantees that the forward while behaves indeed as a while loop. The backward direction is more subtle, and we discuss how it can be interpreted as part of a stateful program in Section 4.

▶ **Lemma 5.** *Whenever $\phi(\boldsymbol{x}) \to \boldsymbol{ax} \prec \boldsymbol{x}$ is provable in* WE-HA$^\omega$*, the following are provable in* WE-HA$^\omega + (\mathtt{whilerec}_\prec)$*:*

$$(\mathtt{while}^+_\prec \phi\,\mathtt{do}\,\boldsymbol{a})\boldsymbol{x} = \mathtt{if}(\phi(\boldsymbol{x}), (\mathtt{while}^+_\prec \phi\,\mathtt{do}\,\boldsymbol{a})(\boldsymbol{ax}), \boldsymbol{x}),$$
$$(\mathtt{while}^-_\prec \phi\,\mathtt{do}\,(\boldsymbol{a}, \boldsymbol{\alpha}))\boldsymbol{xv} = \mathtt{if}(\phi(\boldsymbol{x}), \boldsymbol{\alpha x}\,\big((\mathtt{while}^-_\prec \phi\,\mathtt{do}\,(\boldsymbol{a}, \boldsymbol{\alpha}))(\boldsymbol{ax})\boldsymbol{v}\big), \boldsymbol{v}).$$

We now define a corresponding rule for any decidable relation $\prec$, namely

$$\frac{\exists \boldsymbol{x}\,(P_\forall(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}))\,\langle \boldsymbol{a} \mid \boldsymbol{\alpha}\rangle\,\exists \boldsymbol{x}\,P_\forall(\boldsymbol{x}) \quad \forall \boldsymbol{x}\,(\phi(\boldsymbol{x}) \to \boldsymbol{ax} \prec \boldsymbol{x})}{\exists \boldsymbol{x}\,P_\forall(\boldsymbol{x})\,\langle \mathtt{while}^+_\prec \phi\,\mathtt{do}\,\boldsymbol{a} \mid \mathtt{while}^-_\prec \phi\,\mathtt{do}\,(\boldsymbol{a}, \boldsymbol{\alpha})\rangle\,\exists \boldsymbol{x}\,(P_\forall(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x}))}\,W_\prec$$

The rule $W_\prec$ is closely connected to the wellfounded while rule of Hoare logic, with the difference that the descent condition is now presented separately. The following result (proven in the appendix) connects the rule to the recursor, and bears some relationship to the interpretation of wellfounded induction in [38].

▶ **Theorem 6.** *The while rule $(W_\prec)$ is admissible in* WE-HA$^\omega + \mathrm{I}_\prec + (\mathtt{whilerec}_\prec)$.

Discussion: The interpretation of classical logic and its interaction with $\mathtt{whilerec}_\prec$

It is well known that classical logic can be given a computational interpretation via the Dialectica by first carrying out a negative translation (see [18, Chapter 10]), where the embedding of classical logic into intuitionistic makes use of a number of (semi-intuitionistic) laws governing negations. Expanding DHL with these laws as primitives would allow for a

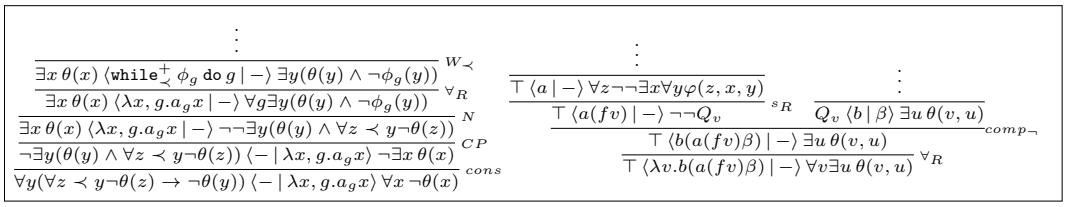

Figure 4 Left: For $g : X \to X$ we use the abbreviation $\phi_g(x) := gx \prec x \wedge \theta(gx)$ and $a_g := \mathtt{while}^+_\prec \phi_g \,\mathtt{do}\, g$. One completes the derivation by using $\exists x(\theta(x) \wedge \phi_g(x)) \,\langle g \mid - \rangle\, \exists y\, \theta(y)$ and $\forall x(\phi_g(x) \to gx \prec x)$, which are easily provable. Right: we use the negative translation to obtain a constructive proof of the double negated statement that appears in the left branch.

streamlined verification of programs extracted from classical proofs, and though detailed exploration of this is beyond the scope of the present paper, we give three examples of the kind of rules we have in mind, in Figure 3.

Here, (CP) and (N) are extremely useful (and reversible) symmetry rules that one often encounters when analysing classical proofs, while $(comp_\neg)$ is a version of composition which illustrates one way in which the negative translation composed with Dialectica eliminates nonconstructive lemmas when proving purely existential formulas, as we discuss below.

An example of how our while rule in particular interacts with these new principles for handling double negation is given on the left side of Figure 4, where we use it to define a realiser for double-negated version of the following *minimum principle*:

$$\exists x\, \theta(x) \vdash \exists x\, (\theta(x) \wedge \forall y \prec x \neg\theta(y))\,,$$

where $\theta(x^X)$ is a quantifier-free formula and $\prec$ is a wellfounded relation on X (we omit simple instances of (ext) to improve readability). This represents a derivation, in a Hoare style system, of the kind of intuitive algorithm for the minimum principle already described in e.g. [6, Section 7], and the imperative flavour of such constructions in relation to the Dialectica interpretation is also discussed in [34], where an explicit connection to learning algorithms is drawn. This general phenomenon of computing approximations to ideal objects via learning is present in almost all approaches to giving a computational meaning to classical reasoning: the epsilon calculus, modified realizability [7], game semantics [6], approaches based on learning [2], the $\lambda\mu$-calculus [32], call-cc [13], classical realizability [22, 24] and others (see [35] for a comparative study of some of these), and we propose that our approach via Hoare logic might help make explicit this intuition in the context of Dialectica.

Another example where we see our rules for classical logic in action is given by the frequently occurring proof pattern where an instance of a nonconstructive principle is eliminated by the Dialectica when used as a lemma in the proof of a $\forall\exists$-theorem. More precisely, suppose that we establish $\forall v \exists u\, \theta(v, u)$ by showing (constructively) that for any v we have $\exists x \forall y\, \varphi(fv, x, y) \to \exists u\, \theta(v, u)$, for some function f, but where the premise is an instance of a principle only provable nonconstructively; for instance, x might be a minimal element in some set, a maximal ideal, or a point after which a sequence is 2^{-fv} stable. Abbreviating $\exists x \forall y\, \varphi(fv, x, y)$ with Q_v, the derivation on the right in Figure 4 provides a basic template for eliminating this nonconstructive principle via $(comp_\neg)$, combining a witness a for the *classical* Dialectica interpretation of the latter with a realizing pair for the *constructive* implication above. There are many cases where the witness a for the double negated lemma could be built in a natural way via using our while loop, against some "counterexample function" provided by the backward witness β. It might then form a core component of a procedural-style program for computing a witness for $\forall v \exists u\, \theta(v, u)$.

4 A procedural language for Dialectica

So far, we have put forward the argument that having a rich library of rules for Dialectica triples, including imperative constructs like the while loop, is beneficial for describing programs naturally arising from mathematics in a procedural way. In this section, we consider this aspect of our system in a more formal manner. For now we consider a restricted system that aligns closely with traditional Hoare logic, comprising a small set of rules for generating "stateful" programs. Because these rules arise from the Dialectica interpretation, the result is a novel procedural language with nonstandard interpretation. To be more precise, we define a language LOOP_D, representing a Dialectica-like extension of a toy procedural language LOOP, and provide both a Hoare logic and an operational semantics, where the latter captures the idea that

$$\mathrm{LOOP}_D = \mathrm{LOOP} + \text{backpropagation},$$

i.e. commands of the language consist of an ordinary forward part, together with a backward component that can be computed via a generalised form of *backpropagation*. We anticipate that this operational semantics that could eventually be extended to our entire system DHL.

In order to define LOOP_D and its corresponding logic, we extend our ambient theory WE-HA$^\omega$ with two new *abstract types* S and T representing two sorts of *state*. The extension of WE-HA$^\omega$ with abstract types is standard in the proof mining literature (see [17]), where those types are typically used to represent abstract spaces from mathematics. We then extend the terms WE-HA$^\omega$ by introducing:

- two equality symbols $=_S$ and $=_T$, which we write as predicates but which will technically be terms $=_S\colon S \to S \to \mathtt{nat}$ and $=_T\colon T \to T \to \mathtt{nat}$ (indicating that equality between states is a decidable property). In addition, we extend all the usual constructs of WE-HA$^\omega$ e.g. quantifiers, recursors, lambda abstraction, so that they apply to our new states;
- two sets of primitive command, the set $\mathtt{Comm}^+$ of *forward commands* of type $S \to S$, and the set $\mathtt{Comm}^-$ of *backward commands* of type $S \to T \to T$;
- a set $\mathtt{Exp}$ of boolean expressions, which are formally a set of decidable predicates on S represented in WE-HA$^\omega$ as characteristic functions of type $S \to S \to \mathtt{nat}$;
- a set $\mathtt{Rel}$ of wellfounded relations on S.

We also assume that we have a set of purely universal axioms $\mathcal{U}$ that characterise the meaning of commands and expressions. We then define the set $\mathtt{Comm}$ of *commands* of LOOP_D as follows, all of which represent new constants or term forming operations in WE-HA$^\omega$ for generating pairs of terms of types $S \to S, S \to T \to T$:

$$C ::= \mathtt{skip} \mid \langle c \,|\, \gamma \rangle \mid C\,;\, C \mid \mathtt{if}\ \phi\ \mathtt{then}\ C\ \mathtt{else}\ C \mid \mathtt{while}_\prec\ \phi\ \mathtt{do}\ C$$

where $c \in \mathtt{Comm}^+$, $\gamma \in \mathtt{Comm}^-$, $\phi \in \mathtt{Exp}$ and $\prec \in \mathtt{Rel}$. The new constructors come equipped with the defining axiom

$$C = \langle C^+ \,|\, C^- \rangle,$$

where $\langle\,|\,\rangle$ is a fixed representation of pairs in WE-HA$^\omega$ and $C^+ : S \to S$ and $C^- : S \to T \to T$ are defined inductively over $\mathtt{Comm}$ as in Figure 5. In particular, notice that by Lemma 5 we have that if $\neg\phi(s)$, then $(\mathtt{while}_\prec\ \phi\ \mathtt{do}\ C_1)s = (\mathtt{skip})s$, and whenever $\phi(s)$ and $C^+ s \prec s$, then

$$(\mathtt{while}_\prec\ \phi\ \mathtt{do}\ C)s = (C\,;\ \mathtt{while}_\prec\ \phi\ \mathtt{do}\ C)s.$$

We denote by WE-HA$^\omega + (\mathrm{LOOP}_D)$ the extension of WE-HA$^\omega$ with all of the above (including the while recursor and well-founded induction axiom for all $\prec \in \mathtt{Rel}$).

LOOP_D	$(\_)^+$	$(\_)^-$
$\mathtt{skip}$	$\lambda s.s$	$\lambda s,t.t$
$\langle c \,|\, \gamma \rangle$	c	γ
$C_1 \,;\, C_2$	$C_2^+ \circ C_1^+$	$C_1^- *_{C_1^+} C_2^-$
$\mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2$	$\lambda s.\mathtt{if}(\phi(s), C_1^+ s, C_2^+ s)$	$\lambda s,t.\mathtt{if}(\phi(s), C_1^- st, C_2^- st)$
$\mathtt{while}_\prec\ \phi\ \mathtt{do}\ C$	$\mathtt{while}^+_\prec\ \phi\ \mathtt{do}\ C^+$	$\mathtt{while}^-_\prec\ \phi\ \mathtt{do}\ (C^+, C^-)$

Figure 5 Decomposition in WE-HA$^\omega$ of LOOP_D terms in System T terms. Remember the Dialectica composition $*$ from Section 3 and the while constructors from Section 3.2.

$$[P]\ \mathtt{skip}\ [P] \qquad \frac{P(s,\gamma st) \to Q(cs,t) \in \mathrm{Ax}}{[P]\,\langle c \,|\, \gamma\rangle\,[Q]} \qquad \frac{P' \to P \quad [P]\,C\,[Q] \quad Q \to Q'}{[P']\,C\,[Q']}$$

$$\frac{[P]\,C_1\,[Q] \quad [Q]\,C_2\,[R]}{[P]\,C_1\,;\,C_2\,[R]} \qquad \frac{[P \wedge \phi]\,C_1\,[R] \quad [Q \wedge \neg\phi]\,C_2\,[R]}{[P \vee_\phi Q]\ \mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2\,[R]} \qquad \frac{[P \wedge \phi]\,C\,[P] \quad \phi(s) \to C^+ s \prec s}{[P]\ \mathtt{while}_\prec\ \phi\ \mathtt{do}\ C\,[P \wedge \neg\phi]}$$

Figure 6 Hoare rules for Dialectica triples in LOOP_D. Notice that we use some abbreviations: $P \wedge \phi$ is sugar for the formula $P(s,t) \wedge \phi(s)$, and $P \vee_\phi Q$ for the formula $P(s,t) \vee_{\phi(s)} Q(s,t)$.

Figure 5 characterises our commands as comprising a standard imperative forward part, with a "Dialectica version" of the same component as the backward part. A corresponding set of specification rules is now given in terms of our Dialectica triples in Figure 6: Here we use the more compact and suggestive abbreviation

$$[P]\,C\,[Q] := \exists s \forall t\, P(s,t)\,\langle C^+ \,|\, C^-\rangle\, \exists s \forall t\, Q(s,t) = \forall s,t\,\big(P(s, C^- st) \to Q(C^+ s, t)\big)$$

and restrict our attention to Dialectica triples whose pre- and postconditions are formulas of the form $\exists s^S \forall t^T\, P(s,t)$ where $P(s,t)$ is quantifier-free, and s,t are the only free variables of type S,T that appear in them. In the second rule, Ax represents some set of axioms that govern the primitive commands, which we assume are included in $\mathcal{U}$. All of these rules are either instances of the main rules from previous sections, or in the case of conditional, easily derivable, and so it is straightforward to prove the following:

▶ **Theorem 7.** *All rules in Figure 6 are admissible in* WE-HA$^\omega$ + (LOOP_D).

We also note that Figure 6 only shows one set of rules corresponding to traditional Hoare logic: Plenty more are admissible (e.g. those for conjunction and disjunction).

An operational semantics for LOOP_D via backpropagation

So far, LOOP_D is just an extension of System T, with equational rules that describe the meaning of terms. We now endow terms of LOOP_D with a big step operational semantics, to highlight how they can be interpreted as "stateful" programs. We first introduce some notation: For any type X we let X^* denote the type of finite sequences of elements of X (while this is not formally part of WE-HA$^\omega$, it can easily be encoded in the system, though we omit details). For $\sigma, \tau : X^*$ we write $\tau :: \sigma : X^*$ for the concatenation of two sequences, and for $x : X$ similarly write $x :: \sigma$ for concatenation with a single element. We write ϵ for the empty list (of any type). Our semantics comprises two components: A forward relation and a backward relation, which we write

$$s, C \Downarrow^+ s', \sigma, \Gamma \quad \text{and} \quad \sigma, \Gamma, t \Downarrow^- \sigma', \Gamma', t'$$

Forward semantics

$$s, \mathtt{skip} \Downarrow^+ s, \epsilon, \epsilon \qquad s, \langle c \,|\, \gamma \rangle \Downarrow^+ cs, [s], [\gamma] \qquad \frac{s, C_1 \Downarrow^+ s', \sigma, \Gamma \quad s', C_2 \Downarrow^+ s'', \sigma', \Gamma'}{s, C_1 \,;\, C_2 \Downarrow^+ s'', \sigma' :: \sigma, \Gamma' :: \Gamma}$$

$$\frac{\phi(s) \quad s, C_1 \Downarrow^+ s', \sigma, \Gamma}{s, \mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2 \Downarrow^+ s', \sigma, \Gamma} \qquad \frac{\neg\phi(s) \quad s, C_2 \Downarrow^+ s', \sigma, \Gamma}{s, \mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2 \Downarrow^+ s', \sigma, \Gamma}$$

$$\frac{\phi(s) \quad s, C \Downarrow^+ s', \sigma, \Gamma \quad s' \prec s \quad s', \mathtt{while}_\prec\, \phi\, \mathtt{do}\, C \Downarrow^+ s'', \sigma', \Gamma'}{s, \mathtt{while}_\prec\, \phi\, \mathtt{do}\, C \Downarrow^+ s'', \sigma' :: \sigma, \Gamma' :: \Gamma} \qquad \frac{\neg\phi(s)}{s, \mathtt{while}_\prec\, \phi\, \mathtt{do}\, C \Downarrow^+ s, \epsilon, \epsilon}$$

Backward semantics

$$\sigma, \Gamma, t \Downarrow^- \sigma, \Gamma, t \qquad s :: \sigma, \gamma :: \Gamma, t \Downarrow^- \sigma, \Gamma, \gamma s t \qquad \frac{\sigma, \Gamma, t \Downarrow^- \sigma', \Gamma', t' \quad \sigma', \Gamma', t' \Downarrow^- \sigma'', \Gamma'', t''}{\sigma, \Gamma, t \Downarrow^- \sigma'', \Gamma'', t''}$$

Figure 7 Operational semantics for LOOP_D.

for $s, s' : S$, $t, t' : T$, $C \in \mathtt{Comm}$, $\sigma, \sigma' : S^*$ and $\Gamma, \Gamma' : (S \to T \to T)^*$. Rules for the semantics are given in Figure 7. If we ignore the stacks, our forward semantics is isomorphic to a standard operational semantics for an imperative language. It is not difficult to show that semantics is sound with respect to the denotational semantics of WE-HA$^\omega$ + (LOOP_D):

▶ **Theorem 8.** *Let C be an arbitrary command in LOOP_D, where we restrict the formation of $\mathtt{while}_\prec\, \phi\, \mathtt{do}\, C_1$ to commands such that $\forall s(\phi(s) \to C_1^+ s \prec s)$ is provable in* WE-HA$^\omega$. *Then for any s there exist s', σ and Γ such that*

$$s, C \Downarrow^+ s', \sigma, \Gamma$$

where $s' = C^+ s$ in WE-HA$^\omega$, *and such that for any $t : T$, σ_0 and Γ_0 there exists t' such that*

$$\sigma :: \sigma_0, \Gamma :: \Gamma_0, t \Downarrow^- \sigma_0, \Gamma_0, t'$$

where $t' = C^- s t$ in WE-HA$^\omega$.

A proof of Theorem 8 can be found in the appendix. Our operational semantics captures the idea that to evaluate a command C in state s, we perform a forward run

$$s, C \Downarrow^+ s_k, [s_{k-1}, \ldots, s_1, s], [\gamma_k, \ldots, \gamma_1]$$

where for each $j = 1, \ldots, k$ we have $s_j = c_j s_{j-1}$ for $\langle c_j \,|\, \gamma_j \rangle$ a primitive command, and $s_k = (c_k \circ \ldots \circ c_1)s = C^+ s$. In other words, each time we are confronted with a primitive command $\langle c_j \,|\, \gamma_j \rangle$ in some intermediate state s_{j-1}, we perform the forward component but push both the intermediate state s_{j-1} and the backward command γ_j onto stacks. The backward run then just pops the intermediate states and commands, with

$$[s_{k-1}, \ldots, s_1, s], [\gamma_k, \ldots, \gamma_1], t \Downarrow^- \epsilon, \epsilon, (\gamma_1 s \circ \ldots \circ \gamma_k s_{k-1})t$$

where $(\gamma_1 s \circ \ldots \circ \gamma_k s_{k-1})t = C^- s t$ In this way, programs in LOOP_D combine a generalised backpropagation algorithm, where in addition to composing functions we can also perform conditionals and loops. The fact that Dialectica is tightly related with manipulations of stacks in a "backward way" has been discovered in [33], therefore it is not a surprise that stacks also play an essential role in our framework.

$$\frac{P_1\,\langle \boldsymbol{a}\,|\,\boldsymbol{\alpha}\rangle\,Q_1 \quad P_2\,\langle \boldsymbol{b}\,|\,\boldsymbol{\beta}\rangle\,Q_2}{P_1 * P_2\,\langle \boldsymbol{a}\,|\,\boldsymbol{\alpha}\rangle \,||\, \langle \boldsymbol{b}\,|\,\boldsymbol{\beta}\rangle\,Q_1 * Q_2} \qquad \frac{P\,\langle \boldsymbol{a}\,|\,\boldsymbol{\alpha}\rangle\,Q}{P * R\,\langle \boldsymbol{a}\,|\,\boldsymbol{\alpha}\rangle \,||\, \mathtt{skip}\; Q * R}$$

Figure 8 Frame-like rules admissible in DHL, $P_1 * P_2\,\langle \boldsymbol{a}\,|\,\boldsymbol{\alpha}\rangle \,||\, \langle \boldsymbol{b}\,|\,\boldsymbol{\beta}\rangle\,Q_1 * Q_2$ is an abbreviation for $P_1 \wedge P_2\,\langle \boldsymbol{a},\boldsymbol{b}\,|\,\boldsymbol{\alpha},\boldsymbol{\beta}\rangle\,Q_1 \wedge Q_2$ *under the condition that* $\boldsymbol{a},\boldsymbol{b}$ *and* $\boldsymbol{\alpha},\boldsymbol{\beta}$ *operate on different variables.*

Discussion: On automatic differentiation

The basic connection between Dialectica and backpropagation is not new: In particular, it was explored in the specific context of differentiation in [16]. We can phrase it within our framework as well: Suppose that our state S represent some space on which we can define functions $c : S \to S$ and we have the notion of a differentials $D_s(c) : S \to S$ at points $s : S$. Suppose that we can also express the reverse differentials $D_s^*(c) : T \to T$ on the dual space T, defined by $D_s^*(c)t := t \circ D_s(c)$. Then for a pair $c_1, c_2 : S \to S$ of differentiable functions, it is well-known that we have the transpose version of the chain rule:

$$D_s^*(c_1) \circ \left(D^*_{c_1(s)}(c_2)\right)$$

or, to put it another way,

$$\langle c_2 \circ c_1\,|\,D^*_{(\cdot)}(c_2 \circ c_1)\rangle = \langle c_1\,|\,D^*_{(\cdot)}(c_1)\rangle\,;\,\langle c_2\,|\,D^*_{(\cdot)}(c_2)\rangle.$$

Therefore, if our primitive commands consist of pairs $\langle c\,|\,\gamma\rangle$ with the property that $\gamma s = D_s^*(c)$, this property is preserved under composition of commands, and the resulting semantics (restricted to composition) gives a version of the traditional backpropagation algorithm.

It should be stressed that our approach is not intended to equate the investigations around Dialectica and differentiation as done in [16], where among other things a precise translation into the differential λ-calculus is provided. In a certain sense, our presentation is orthogonal, presenting a general backpropagation procedure connected to imperative commands. Actually, for commands of the form $\langle c\,|\,D^*_{(\cdot)}(c)\rangle$ to have any real place within our framework, they have to connect to the overarching logic: For what kind of predicates P, Q on spaces S and their duals T is it naturally the case that $\forall s^S, t^{S\to R}\,(P(s, t \circ D_s(c)) \to Q(cs, t))$ holds? This is currently unclear to the authors.

5 Conclusion and Future Work

Towards a genuine imperative language, concurrency. So far, our presentation of LOOP_D gives no concrete *data structure* on the states S, T above, and our primitive commands c, γ are completely abstract. This is the only missing step in order transform LOOP_D into a full imperative language with backpropagation. This can be achieved, for example, by introducing, as constants in our formal system, a countable collection of state variables $\mathtt{x}_1, \mathtt{x}_2, \ldots$ modeling a memory heap for the state S, which would be now seen as a partial function mapping state variables to nat, and similarly we could ask for a further memory heap for the state T. In a similar way we could introduce arithmetic expressions, and include an assignment operation in our primitive commands. However, we leave the details to future work.

Another feature that seems to naturally appear in our framework is *concurrency*, where in particular the rules we give in Figure 8 are admissible in DHL. The interpretation of the conclusion of the left hand rule is

$$\forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{v}\,\left(|P_1|^{\boldsymbol{x}}_{\boldsymbol{\alpha xu}} \wedge |P_2|^{\boldsymbol{y}}_{\boldsymbol{\beta yv}} \to |Q_1|^{\boldsymbol{ax}}_{\boldsymbol{u}} \wedge |Q_2|^{\boldsymbol{by}}_{\boldsymbol{v}}\right)$$

i.e. Dialectica realisers act on *independent* variables. Accordingly, we also have the basic variant of the frame rule in the right of Figure 8. A proper incorporation of concurrency into Dialectica would be extremely interesting, making the kind local reasoning already implicit in Dialectica more formal. Strictly related with concurrency would be to extend Dialectica to bunched logic [29]. In fact, success here through our approach as Hoare Logic, would lead to further developments in the direction of its extension to separation logic [37] (or intermediate logics [3]). The existence of Dialectica interpretations of linear logic [31] demonstrate the possibility of handling the computational content of substructural logics with Dialectica.

Case studies in probability. Over the last few years, proof mining has been rapidly expanding to probability theory. An entirely different route for expansion is then to investigate Dialectica for probabilistic programs, in order to make formal the extraction processes behind those new works. Here we could potentially combine our system for Dialectica with variants of Hoare logic for probabilistic programs, e.g. [9]. We anticipate that this is a fertile new territory for interesting algorithms: Here, iterative trial-and-error algorithms seem fundamental, with several different forms of probabilistic convergence represented computationally in terms of learning procedures, which, informally speaking, test elements of a stochastic processes until a region is found which is locally stable with some sufficiently high probability (see [26] for a more detailed account of this phenomenon in the specific context of stochastic convergence, and [27, 28] for examples of recent, relevant case studies, which in turn build on earlier work in ergodic theory [5]). We note that even elementary facts from probability have resulted in algorithms of extreme complexity, such as the analysis of Egorov's theorem in [4], and propose that the procedural paradigm explored in this paper might be well suited to describing and simplifying such algorithms.

Operational semantics of Dialectica. Finally, it would be interesting to try to extend the operational semantics of LOOP_D terms to the more general class of realisers handled by the full system DHL. Here, of course, we have to contend with full functional programming, but we conjecture that e.g. through a sensible use of monads one could potentially characterise general Dialectica realisers more dynamically, by describing in more detail how the forward and backward directions interact. This perspective brings us closer to machine-based interpretations of proofs such as classical realizability [22], whose applicability to witness extraction from classical proofs as discussed in [24] is undoubtedly relevant, given the association with negative translations. The starting point would surely be the already mentioned [33], where a fundamental connection between Dialectica and the KAM is given.

References

1 Wilhelm Ackermann. Begründung des "tertium non datur" mittels der Hilbertschen Theorie der Widerspruchsfreiheit. *Mathematische Annalen*, 93:1–36, 1924. doi:10.1007/BF01449946.

2 Federico Aschieri and Stefano Berardi. Interactive learning-based realizability for Heyting arithmetic with EM1. *Logical Methods in Computer Science*, 6(3), 2010. doi:10.2168/LMCS-6(3:19)2010.

3 Federico Aschieri, Adata Ciabattoni, and Francesco Genco. On the concurrent computational content of intermediate logics. *Theoretical Computer Science*, 813:375–409, 2020. doi:10.1016/j.tcs.2020.01.022.

4 Jeremy Avigad, Edward T. Dean, and Jason Rute. A metastable dominated convergence theorem. *Journal of Logic & Analysis*, 4(3):1–19, 2012. doi:10.4115/jla.2012.4.3.

5 Jeremy Avigad, Philipp Gerhardy, and Henry Towsner. Local stability of ergodic averages. *Transactions of the American Mathematical Society*, 362(1):261–288, 2010. doi:10.1090/S0002-9947-09-04814-4.

6 Stefano Berardi, Marc Bezem, and Thierry Coquand. On the computational content of the axiom of choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998. doi:10.2307/2586854.

7 Ulrich Berger and Helmut Schwichtenber. Program extraction from classical proofs. In *Logic and Computational Complexity workshop (LCC'94)*, volume 960 of *Lecture Notes in Computer Science*, pages 77–97, 1995. doi:10.1007/3-540-60178-3_80.

8 Valeria de Paiva. *The Dialectica categories*. PhD thesis, University of Cambridge, 1991. Published as Technical Report 213, Computer Laboratory, University of Cambridge. doi:10.48456/tr-213.

9 Jerry den Hartog and Erik P. de Vink. Verifying probabilistic programs using a Hoare like logic. *International Journal of Foundations of Computer Science*, 13:315–340, 2002. doi:10.1142/S012905410200114X.

10 Justus Diller. Eine Variante zur Dialectica-Interpretation der Heyting-Arithmetik endlicher Typen. *Archiv für mathematische Logik und Grundlagenforschung*, 16(1–2):49–66, 1974. doi:10.1007/BF02025118.

11 Philipp Gerhardy and Ulrich Kohlenbach. General logical metatheorems for functional analysis. *Transactions of the American Mathematical Society*, 360:2615–2660, 2008. doi:10.1090/S0002-9947-07-04429-7.

12 Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *dialectica*, 12(3–4):280–287, 1958. doi:10.1111/j.1746-8361.1958.tb01464.x.

13 Timothy Griffin. A formulae-as-type notion of control. In *Proceedings of Principles of Programming Languages (POPL'90)*, pages 47–58. ACM, 1990. doi:10.1145/96709.96714.

14 Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969. doi:10.1145/363235.363259.

15 William A. Howard. Hereditarily majorizable functionals of finite type. In Anne S. Troelstra, editor, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, 1973. doi:10.1007/BFb0066739.

16 Marie Kerjean and Pierre-Marie Pédrot. δ is for Dialectica. In *Proceedings of Logic in Computer Science (LICS'24)*, pages 48:1–13. ACM, 2024. doi:10.1145/3661814.3662106.

17 Ulrich Kohlenbach. Some logical metatheorems with applications in functional analysis. *Transactions of the American Mathematical Society*, 357(1):89–128, 2005. doi:10.1090/s0002-9947-04-03515-9.

18 Ulrich Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Springer Monographs in Mathematics. Springer, 2008. doi:10.1007/978-3-540-77533-1.

19 Ulrich Kohlenbach. Recent progress in proof mining in nonlinear analysis. *IFCoLoG Journal of Logics and their Applications*, 10(4):3361–3410, 2017. URL: https://www2.mathematik.tu-darmstadt.de/~kohlenbach/progress-new.pdf.

20 Ulrich Kohlenbach. Proof-theoretic methods in nonlinear analysis. In *Proceedings of the International Congress of Mathematicians 2018*, volume 2, pages 61–82. World Scientific, 2019. doi:10.1142/9789813272880_0045.

21 Ulrich Kohlenbach. Local formalizations in nonlinear analysis and related areas and proof-theoretic tameness. In *Kreisel's Interests: On the Foundations of Logic and Mathematics*, volume 41 of *Tributes*, pages 45–61. College Publications, 2020. URL: https://www2.mathematik.tu-darmstadt.de/~kohlenbach/Kreisel2018.pdf.

22 Jean-Louis Krivine. Realizability in classical logic. in: Interactive models of computation and program behaviour. *Panoramas et synthéses*, 27:197–229, 2009. URL: https://hal.science/hal-00154500/document.

23 Wilfried Meyer-Viol. *Instantial Logic*. PhD thesis, University of Amsterdam, 1985. URL: https://eprints.illc.uva.nl/id/eprint/1984/2/DS-1995-11.text.pdf.

24 Alexandre Miquel. Existential witness extraction in classical realizability and via a negative translation. *Logical Methods in Computer Science*, 7(2:2):1–47, 2011. doi:10.2168/LMCS-7(2:2)2011.

25 Morenikeji Neri and Nicholas Pischke. Proof mining and probability theory. *Forum of Mathematics, Sigma*, 13:e187, 2025. doi:10.1017/fms.2025.10138.

26 Morenikeji Neri, Nicholas Pischke, and Thomas Powell. Generalized learnability of stochastic principles. In *Proceedings of Computability in Europe (CiE'25)*, volume 15764 of *LNCS*, pages 333–348, 2025. doi:10.1007/978-3-031-95908-0_24.

27 Morenikeji Neri and Thomas Powell. On quantitative convergence for stochastic processes: Crossings, fluctuations and martingales. *Transactions of the American Mathematical Society, Series B*, 12:974–1019, 2025. doi:10.1090/btran/229.

28 Morenikeji Neri and Thomas Powell. A quantitative Robbins-Siegmund theorem. *Annals of Applied Probability*, 2025. To appear. doi:10.48550/arXiv.2410.15986.

29 Peter O'Hearn and David Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999. doi:10.2307/421090.

30 Paulo Oliva. Unifying functional interpretations. *Notre Dame Journal of Formal Logic*, 47(2):263–290, 2006. doi:10.1305/ndjfl/1153858651.

31 Paulo Oliva. Functional interpretations of linear and intuitionistic logic. *Information and Computation*, 208(5):565–577, 2010. doi:10.1016/j.ic.2008.11.008.

32 Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of Logic Programming and Automated Reasoning (LPAR'92)*, pages 190–201. Springer Berlin Heidelberg, 1992. doi:10.1007/BFb0013061.

33 Pierre-Marie Pédrot. A functional functional interpretation. In *Joint proceedings of Computer Science Logic and Logic in Computer Science (CSL–LICS'14)*, pages 77:1–10. ACM, 2014. doi:10.1145/2603088.2603094.

34 Thomas Powell. Gödel's functional interpretation and the concept of learning. In *Proceedings of Logic in Computer Science (LICS '16)*, pages 136–145. ACM, 2016. doi:10.1145/2933575.2933605.

35 Thomas Powell. Computational interpretations of classical reasoning: From the epsilon calculus to stateful programs. In *Mathesis Universalis, Computability and Proof*, volume 412 of *Synthese Library*, pages 255–290. Springer, 2019. doi:10.1007/978-3-030-20447-1_14.

36 Thomas Powell. Proofs as stateful programs: A first-order logic with abstract Hoare triples. *Logical Methods in Computer Science*, 20(1):7:1–7:32, 2024. doi:10.46298/LMCS-20(1:7)2024.

37 John C. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings of Logic in Computer Science (LICS'02)*, pages 55–74. IEEE, 2002. doi:10.1109/LICS.2002.1029817.

38 Helmut Schwichtenberg. Dialectica interpretation of well-founded induction. *Mathematical Logic Quarterly*, 54(3):229–239, 2008. doi:10.1002/malq.200710045.

39 Davide Trotta, Jonathan Weinberger, and Valeria de Paiva. Skolem, Gödel, and Hilbert fibrations. available at https://arxiv.org/abs/2407.15765, 2024.

A Appendix

All proofs contained in the appendix are routine, involving a standard structural induction with multiple cases.

Proof of Theorem 3. The proof follows the standard soundness theorem for the Dialectica, so we only give details of representative and interesting cases.

- The axioms are admissible by definition.
- For $(p\wedge_R)$, the premise is

$$\forall \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}\left(|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha x v w}} \to |Q|^{\boldsymbol{ax}}_{\boldsymbol{v}} \wedge |R|^{\boldsymbol{by}}_{\boldsymbol{w}}\right)$$

and therefore we have

$$\forall \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{w} \left(|P|^{\boldsymbol{x}}_{\tilde{\boldsymbol{\alpha}}\boldsymbol{x}\boldsymbol{w}\boldsymbol{v}} \to |R|^{\boldsymbol{b}\boldsymbol{y}}_{\boldsymbol{w}} \wedge |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \right)$$

for $\tilde{\boldsymbol{\alpha}}\boldsymbol{x}\boldsymbol{w}\boldsymbol{v} := \boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}\boldsymbol{w}$. All other basic actions are proved similarly.

- ($cond_R$) is the more interesting of the conditionals. Here, we have

$$\forall \boldsymbol{x}, \boldsymbol{v} \left(|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \to |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \right)$$

and

$$\forall \boldsymbol{x}, \boldsymbol{w} \left(|P|^{\boldsymbol{x}}_{\boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w}} \to |R|^{\boldsymbol{b}\boldsymbol{x}}_{\boldsymbol{w}} \right).$$

Define $\boldsymbol{\gamma} := \lambda \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}.\mathtt{if}(|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}}, \boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w}, \boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v})$. We need to prove that

$$\forall \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w} \left(|P|^{\boldsymbol{x}}_{\boldsymbol{\gamma}\boldsymbol{x}\boldsymbol{v}\boldsymbol{w}} \to |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \wedge |R|^{\boldsymbol{b}\boldsymbol{x}}_{\boldsymbol{w}} \right)$$

There are two possibilities: Fixing $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}$, either $|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}}$ holds, in which case

$$|P|^{\boldsymbol{x}}_{\boldsymbol{\gamma}\boldsymbol{x}\boldsymbol{v}\boldsymbol{w}} \to |P|^{\boldsymbol{x}}_{\boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w}} \to |P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \wedge |P|^{\boldsymbol{x}}_{\boldsymbol{\beta}\boldsymbol{x}\boldsymbol{w}}$$

and so the conclusion is true, or $\neg |P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}}$, and then $|P|^{\boldsymbol{x}}_{\boldsymbol{\gamma}\boldsymbol{x}\boldsymbol{v}\boldsymbol{w}} \to |P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \to \bot$, and so the result automatically follows by ex-falso-quodlibet.

- (imp) and (exp) are immediate.
- ($comp$) is standard and fundamental to Dialectica: If

$$\forall \boldsymbol{x}, \boldsymbol{v} \left(|P|^{\boldsymbol{x}}_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \to |Q|^{\boldsymbol{a}\boldsymbol{x}}_{\boldsymbol{v}} \right)$$

and

$$\forall \boldsymbol{u}, \boldsymbol{w} \left(|Q|^{\boldsymbol{u}}_{\boldsymbol{\beta}\boldsymbol{u}\boldsymbol{w}} \to |R|^{\boldsymbol{b}\boldsymbol{u}}_{\boldsymbol{w}} \right)$$

then fixing $\boldsymbol{x}, \boldsymbol{w}$, setting $\boldsymbol{u} := \boldsymbol{a}\boldsymbol{x}$ and $\boldsymbol{v} := \boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{w}$ gives the desired result.

- The first four quantifier rules are also standard, though a little care is needed if we allow them to apply to tuples. The most involved in ($\forall_R$). If

$$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P|^{\boldsymbol{y}}_{\boldsymbol{\alpha}\boldsymbol{y}\boldsymbol{v}} \to |Q(\boldsymbol{x})|^{\boldsymbol{a}\boldsymbol{y}}_{\boldsymbol{v}} \right)$$

then

$$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P|^{\boldsymbol{y}}_{\boldsymbol{\alpha}\boldsymbol{y}\boldsymbol{v}} \to |Q(x_1, \ldots, x_n)|^{(\lambda x_n . \boldsymbol{a}\boldsymbol{y})x_n}_{\boldsymbol{v}} \right)$$

which is just

$$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P|^{\boldsymbol{y}}_{\boldsymbol{\alpha}\boldsymbol{y}\boldsymbol{v}} \to |\forall x_n \, Q(x_1, \ldots, x_n)|^{\lambda x_n . \boldsymbol{a}\boldsymbol{y}}_{x_n, \boldsymbol{v}} \right)$$

and continuing for the rest of the tuple we obtain

$$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P|^{\boldsymbol{y}}_{\boldsymbol{\alpha}\boldsymbol{y}\boldsymbol{v}} \to |\forall \boldsymbol{x} \, Q(\boldsymbol{x})|^{\lambda \boldsymbol{x} . \boldsymbol{a}\boldsymbol{y}}_{\boldsymbol{x}, \boldsymbol{v}} \right)$$

which is just

$$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P|^{\boldsymbol{y}}_{(\lambda \boldsymbol{y}, \boldsymbol{x} . \boldsymbol{\alpha}\boldsymbol{y})\boldsymbol{y}\boldsymbol{x}\boldsymbol{v}} \to |\forall \boldsymbol{x} \, Q(\boldsymbol{x})|^{(\lambda \boldsymbol{y}, \boldsymbol{x} . \boldsymbol{a}\boldsymbol{y})\boldsymbol{y}}_{\boldsymbol{x}, \boldsymbol{v}} \right)$$

where here we note that the condition $\boldsymbol{x}$ not free in P ensures that the free variables of $|P \to \forall \boldsymbol{x} \, Q(\boldsymbol{x})|^{\boldsymbol{f}, \boldsymbol{F}}_{\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{v}}$ are the same as those of $P \to \forall \boldsymbol{x} \, Q(\boldsymbol{x})$.

- (s_L) and (s_R) follow in a straightforward way from the usual quantifier rules, and the conclusion and premise is identical for (ϵ_R) and (ϵ_L).
- $(cons)$ is immediate, and (exp) follows from the rule of extensionality in WE-HA$^\omega$.
- As usual, (ind) is proven by induction. We have

  $$\forall \boldsymbol{y}, \boldsymbol{v} \left(|P(x)|^{\boldsymbol{y}}_{\boldsymbol{\alpha}(x)\boldsymbol{y}\boldsymbol{v}} \to |P(x+1)|^{\boldsymbol{a}(x)\boldsymbol{y}}_{\boldsymbol{v}} \right)$$

  for all $x : \mathtt{nat}$, so fixing $\boldsymbol{y}$, $\boldsymbol{v}$ and defining $\boldsymbol{b} := \mathtt{rec}\, \boldsymbol{a}\boldsymbol{y}$ and $\boldsymbol{\beta}$ as in Section 3, we prove by induction that

  $$|P(x-z)|^{\boldsymbol{b}(x-z)}_{\boldsymbol{\beta}z} \to |P(x)|^{\boldsymbol{b}x}_{\boldsymbol{v}}$$

  for $z = 0, \ldots, x$. The base case is immediate, and for the induction step we use that $|P(x-z-1)|^{\boldsymbol{b}(x-z-1)}_{\boldsymbol{\beta}(z+1)}$ is equivalent to

  $$|P(x-z-1)|^{\boldsymbol{b}(x-z-1)}_{\boldsymbol{\alpha}(x-z-1)(\boldsymbol{b}(x-z-1))(\boldsymbol{\beta}z)}$$

  which by the premise of the rule allows us to obtain

  $$|P(x-z)|^{\boldsymbol{a}(x-z-1)(\boldsymbol{b}(x-z-1))}_{\boldsymbol{\beta}z}$$

  which is just $|P(x-z)|^{\boldsymbol{b}(x-z)}_{\boldsymbol{\beta}z}$. Thus we can apply the induction hypothesis. For $x := z$ we then have

  $$|P(0)|^{\boldsymbol{y}}_{\boldsymbol{\beta}x} \to |P(x)|^{\boldsymbol{b}x}_{\boldsymbol{v}}$$

  and the result follows by definition.

The remaining cases are straightforward. ◀

Proof of Theorem 4. We refer to the axiomatisation given in [18, Section 3]. We first note that, in our system, provability of $\top \vdash P \to Q$ is equivalent to provability of $P \vdash Q$. With that in mind, for the axioms of intuitionistic logic, both contraction axioms follow from the conditional rules, while weakening, permutation, and ex falso quodlibet are clearly derivable. The quantifier axioms follow from (s_R) and (s_L). Both modus ponens and syllogism are instances of $(comp)$, where for the former we note that if $\top \vdash P$ and $\top \vdash P \to Q$, then also $P \vdash Q$, and thus $\top \vdash Q$. Exportation and importation are identical in both systems, while expansion is provable using a combination of the rules for $\vee$. The quantifier rules are just $(\exists_L)$ and $(\forall_R)$. For arithmetic: We assume that the axioms and rules for equality and System T are included in Ax, and the quantifier-free rules of extensionality included in Rule, so all of these are then provable in our system. Replacing the induction axioms with the equivalent rule, it is not hard to show that the latter is derivable from (ind). ◀

Proof of Theorem 6. Suppose that the premises of the rule hold, and so in particular by the left hand premise

$$\forall \boldsymbol{x}, \boldsymbol{v}\, (|P_\forall(\boldsymbol{x})|_{\boldsymbol{\alpha}\boldsymbol{x}\boldsymbol{v}} \wedge \phi(\boldsymbol{x}) \to |P_\forall(\boldsymbol{a}\boldsymbol{x})|_{\boldsymbol{v}})$$

Writing $\boldsymbol{b} := \mathtt{while}^+_\prec\, \phi\, \mathtt{do}\, \boldsymbol{a}$ and $\boldsymbol{\beta} := \mathtt{while}^-_\prec\, \phi\, \mathtt{do}\, (\boldsymbol{a}, \boldsymbol{\alpha})$, we are done if we can prove $\forall \boldsymbol{x}\, A(\boldsymbol{x})$ for

$$A(\boldsymbol{x}) := \forall \boldsymbol{v} \left(|P_\forall(\boldsymbol{x})|_{\boldsymbol{\beta}\boldsymbol{x}\boldsymbol{v}} \to |P_\forall(\boldsymbol{b}\boldsymbol{x})|_{\boldsymbol{v}} \wedge \neg\phi(\boldsymbol{b}\boldsymbol{x}) \right)$$

Propositional rules: Axioms, basic actions, conditionals, switching, composition.

$$\bot \langle \boldsymbol{a} \mid - \rangle P \qquad P_\exists \langle - \mid \boldsymbol{\alpha} \rangle \top \qquad P_\exists \langle \lambda \boldsymbol{x}.\boldsymbol{x} \mid - \rangle P_\exists \qquad \frac{P_\exists \to Q_\forall \in \mathrm{Ax}}{P_\exists \langle - \mid - \rangle Q_\forall} \qquad \frac{P_\exists \langle - \mid - \rangle Q_\forall}{P'_\exists \langle - \mid - \rangle Q'_\forall} \text{ for } \frac{P_\exists \to Q_\forall}{P'_\exists \to Q'_\forall} \in \mathrm{Rule}$$

$$\frac{P_\exists \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle Q \wedge R}{P_\exists \langle \boldsymbol{b}, \boldsymbol{a} \mid - \rangle R \wedge Q}\, {}^{p\wedge_R} \qquad \frac{P_\exists \wedge Q_\exists \langle \boldsymbol{a} \mid - \rangle R}{Q_\exists \wedge P_\exists \langle \tilde{\boldsymbol{a}} \mid - \rangle R}\, {}^{p\wedge_L} \qquad \frac{P_\exists \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle Q \vee_c R}{P_\exists \langle \boldsymbol{b}, \boldsymbol{a} \mid - \rangle R \vee_{\tilde{c}} Q}\, {}^{p\vee_R} \qquad \frac{P_\exists \vee_c Q_\exists \langle \boldsymbol{a} \mid - \rangle R}{Q_\exists \vee_{\tilde{c}} P_\exists \langle \tilde{\boldsymbol{a}} \mid - \rangle R}\, {}^{p\vee_L}$$

$$\frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q}{P_\exists \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle Q \vee_0 R}\, {}^{\vee_R} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q}{P_\exists \wedge R_\exists \langle \boldsymbol{a}_\pi \mid - \rangle Q}\, {}^{\wedge_L} \qquad \frac{P_\exists \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle Q \wedge R}{P_\exists \langle \boldsymbol{a} \mid - \rangle Q}\, {}^{\wedge_R} \qquad \frac{P_\exists \vee_0 R_\exists \langle \boldsymbol{a} \mid - \rangle Q}{P_\exists \langle \boldsymbol{a}_p \mid - \rangle Q}\, {}^{\vee_L}$$

$$\frac{P_\exists \wedge \phi \langle \boldsymbol{a} \mid - \rangle R \quad Q_\exists \wedge \neg\phi \langle \boldsymbol{b} \mid - \rangle R}{P_\exists \vee_\phi Q_\exists \langle \lambda \boldsymbol{x}, \boldsymbol{y}.\mathtt{if}(\phi, \boldsymbol{a}\boldsymbol{x}, \boldsymbol{b}\boldsymbol{y}) \mid - \rangle R}\, {}^{cond_L} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q \quad P_\exists \langle \boldsymbol{b} \mid - \rangle R}{P_\exists \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle Q \wedge R}\, {}^{cond_R}$$

$$\frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q_\exists \to R}{P_\exists \wedge Q_\exists \langle \boldsymbol{a} \mid - \rangle R}\, {}^{imp} \qquad \frac{P_\exists \wedge Q_\exists \langle \boldsymbol{a} \mid - \rangle R}{P_\exists \langle \boldsymbol{a} \mid - \rangle Q_\exists \to R}\, {}^{exp} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q_\exists \quad Q_\exists \langle \boldsymbol{b} \mid - \rangle R}{P_\exists \langle \boldsymbol{b} \circ \boldsymbol{a} \mid - \rangle R}\, {}^{comp}$$

Quantifier rules: Term introduction, λ-abstraction and application, epsilon terms.

$$\frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q(\boldsymbol{t})}{P_\exists \langle \lambda\_.t, \boldsymbol{a} \mid - \rangle \exists \boldsymbol{x}\, Q(\boldsymbol{x})}\, {}^{\exists_R}$$

$$\frac{P_\exists(\boldsymbol{x}) \langle \boldsymbol{a} \mid - \rangle Q}{\exists \boldsymbol{x}\, P_\exists(\boldsymbol{x}) \langle \lambda \boldsymbol{x}.\boldsymbol{a} \mid - \rangle Q}\, {}^{\exists_L} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid - \rangle Q(\boldsymbol{x})}{P_\exists \langle \lambda \boldsymbol{y}, \boldsymbol{x}.\boldsymbol{a}\boldsymbol{y} \mid - \rangle \forall \boldsymbol{x}\, Q(\boldsymbol{x})}\, {}^{\forall_R} \qquad \boldsymbol{x} \text{ not free in } Q \text{ resp. } P \text{ for } \exists_L \text{ resp. } \forall_R$$

$$\frac{\exists \boldsymbol{x}\, P_\exists(\boldsymbol{x}) \langle \boldsymbol{a} \mid - \rangle Q}{P_\exists(\boldsymbol{t}) \langle \boldsymbol{a}\boldsymbol{t} \mid - \rangle Q}\, {}^{s_L} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid - \rangle \forall \boldsymbol{x}\, Q(\boldsymbol{x})}{P_\exists \langle \lambda \boldsymbol{y}.\boldsymbol{a}\boldsymbol{y}\boldsymbol{t} \mid - \rangle Q(\boldsymbol{t})}\, {}^{s_R} \qquad \frac{P_{qf} \langle \boldsymbol{a}, \boldsymbol{b} \mid - \rangle \exists \boldsymbol{x}\, Q(\boldsymbol{x})}{P_{qf} \langle \boldsymbol{b} \mid - \rangle Q(\boldsymbol{a})}\, {}^{\epsilon_R}$$

Consequence, extensionality, and induction/recursion

$$\frac{P'_\exists \to_D P_\exists \quad P_\exists \langle \boldsymbol{a} \mid - \rangle Q \quad Q \to_D Q'}{P'_\exists \langle \boldsymbol{a} \mid - \rangle Q'}\, {}^{cons} \qquad \frac{P_\exists \langle \boldsymbol{a} \mid \boldsymbol{\alpha} \rangle Q \quad \boldsymbol{a}, \boldsymbol{\alpha} = \boldsymbol{b}, \boldsymbol{\beta}}{P_\exists \langle \boldsymbol{b} \mid \boldsymbol{\beta} \rangle Q}\, {}^{ext} \qquad \frac{P_\exists(x) \langle \boldsymbol{a}x \mid - \rangle P_\exists(x+1)}{P_\exists(0) \langle \mathtt{rec}\, \boldsymbol{a} \mid - \rangle \forall x\, P_\exists(x)}\, {}^{ind}$$

Figure 9 Simplified rules for Dialectica triples with empty backward realiser.

We do this using the wellfounded induction rule $\mathrm{I}_\prec$. So fixing $\boldsymbol{x}$, we assume that $A(\boldsymbol{y})$ holds for all $\boldsymbol{y} \prec \boldsymbol{x}$. To prove $A(\boldsymbol{x})$, we use Lemma 5, which is applicable for any $\boldsymbol{x}$ thanks to the right hand premise of $(W_\prec)$. There are two cases to consider. If $\neg\phi(\boldsymbol{x})$ then $\boldsymbol{b}\boldsymbol{x} = \boldsymbol{x}$ and $\boldsymbol{\beta}\boldsymbol{x} = \lambda \boldsymbol{v}.\boldsymbol{v}$, and then $A(\boldsymbol{x})$ becomes

$$\forall \boldsymbol{v}\, (|P_\forall(\boldsymbol{x})|_{\boldsymbol{v}} \to |P_\forall(\boldsymbol{x})|_{\boldsymbol{v}} \wedge \neg\phi(\boldsymbol{x}))$$

which is provable in this case. On the other hand, if $\phi(\boldsymbol{x})$ then we have $\boldsymbol{b}\boldsymbol{x} = \boldsymbol{b}(\boldsymbol{a}\boldsymbol{x})$, $\boldsymbol{\beta}\boldsymbol{x} = \lambda \boldsymbol{v}.\boldsymbol{\alpha}\boldsymbol{x}(\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{v})$ and so $A(\boldsymbol{x})$ becomes

$$\forall \boldsymbol{v}\, \Big(|P_\forall(\boldsymbol{x})|_{\boldsymbol{\alpha}\boldsymbol{x}(\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{v})} \to |P_\forall(\boldsymbol{b}(\boldsymbol{a}\boldsymbol{x}))|_{\boldsymbol{v}} \wedge \neg\phi(\boldsymbol{b}(\boldsymbol{a}\boldsymbol{x}))\Big)$$

But from $\phi(\boldsymbol{x})$ we also obtain $\boldsymbol{a}\boldsymbol{x} \prec \boldsymbol{x}$, and so by the induction hypothesis we can assume

$$\forall \boldsymbol{v}\, \Big(|P_\forall(\boldsymbol{a}\boldsymbol{x})|_{\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{v}} \to |P_\forall(\boldsymbol{b}(\boldsymbol{a}\boldsymbol{x}))|_{\boldsymbol{v}} \wedge \neg\phi(\boldsymbol{b}(\boldsymbol{a}\boldsymbol{x}))\Big)$$

It suffices therefore to show that

$$\forall \boldsymbol{v}\, \Big(|P_\forall(\boldsymbol{x})|_{\boldsymbol{\alpha}\boldsymbol{x}(\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{v})} \to |P_\forall(\boldsymbol{a}\boldsymbol{x})|_{\boldsymbol{\beta}(\boldsymbol{a}\boldsymbol{x})\boldsymbol{v}}\Big)$$

and this is immediate from the left hand premise of $(W_\prec)$ and the fact that $\phi(\boldsymbol{x})$ holds, and so we have completed the induction step and therefore the proof. ◀

Proof of Theorem 8. For `skip` and the primitive commands this is immediate. The core of the proof lies in the composition rule. Here, for any s, by the induction hypothesis there exist s', σ', Γ' and s'', σ', Γ' such that

$$\begin{cases} s, C_1 \Downarrow^+ s', \sigma, \Gamma \\ s', C_2 \Downarrow^+ s'', \sigma', \Gamma' \end{cases}$$

and therefore

$$s, C_1 \,;\, C_2 \Downarrow^+ s'', \sigma' :: \sigma, \Gamma' :: \Gamma$$

for $s' = C_1^+ s$ and $s'' = C_2^+ s' = (C_2^+ \circ C_1^+)s = (C_1 \,;\, C_2)^+ s$. For the backward direction, again using the induction hypothesis, for any t, σ_0 and Γ_0 we have

$$\begin{cases} \sigma' :: \sigma :: \sigma_0, \Gamma' :: \Gamma :: \Gamma_0, t \Downarrow^- \sigma :: \sigma_0, \Gamma :: \Gamma_0, t' \\ \sigma :: \sigma_0, \Gamma :: \Gamma_0, t' \Downarrow^- \sigma_0, \Gamma_0, t'' \end{cases}$$

and therefore

$$\sigma' :: \sigma :: \sigma_0, \Gamma' :: \Gamma :: \Gamma_0, t \Downarrow^- \sigma_0, \Gamma_0, t''$$

for $t' = C_2^- s' t$ and $t'' = C_1^- s t' = C_1^- s(C_2^- s' t) = C_1^- s(C_2^-(C_1^+ s)t) = (C_1 \,;\, C_2)^- s t$. The conditionals are straightforward, since whenever $\phi(s)$ then in WE-HA$^\omega$ we have

$$\begin{aligned} (\mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2)^+ s &= C_1^+ s \\ (\mathtt{if}\ \phi\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2)^- s t &= C_1^- s t \end{aligned}$$

and similarly for $\neg\phi(s)$. Finally, for the while loop we use induction on $\prec$, using that if $\phi(s)$ then $(\mathtt{while}_\prec\ \phi\ \mathtt{do}\ C)s = (C \,;\, \mathtt{while}_\prec\ \phi\ \mathtt{do}\ C)$ (and also $(\mathtt{while}_\prec\ \phi\ \mathtt{do}\ C)s = (\mathtt{skip})s$ if $\neg\phi(s)$), and so the induction step is essentially the same as the composition rule. ◀